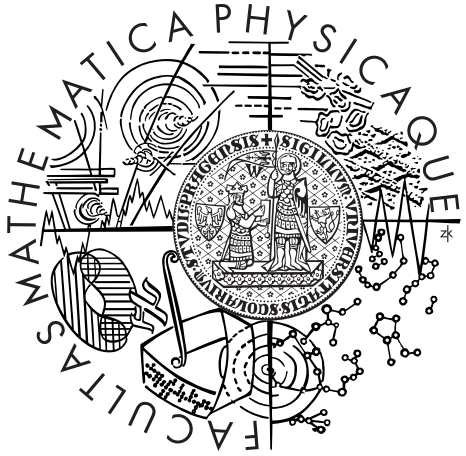
Charles University in Prague
Faculty of Mathematics and Physics

MASTER THESIS



Bc. Lukáš Navrátil

The Impact of Image Resolution on the Precision of Content-based Retrieval

Department of Software Engineering, MFF UK

Supervisor of the master thesis: RNDr. Jakub Lokoč, Ph.D.

Study programme: Informatics

Specialization: Software systems

Prague 2015

I would like to thank my supervisor, Jakub Lokoč, for numerous advice he has given me. I would also like to thank my colleagues from our software project, mainly Přemysl Čech and Tomáš Grušup, for creating the multimedia exploration framework. Last but not least, please allow me to thank to Eva Vopátková for helping me with the language correction.

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In Prague, 30th July 2015 Lukáš Navrátil

Název práce: Vliv rozlišení obrázku na přesnost vyhledávání podle obsahu

Autor: Lukáš Navrátil

Katedra: Katedra softwarového inženýrství

Vedoucí diplomové práce: RNDr. Jakub Lokoč, Ph.D., Katedra softwarového inženýrství

Abstrakt: Tato práce se zaměřuje na porovnání metod pro podobnostní vyhledávání obrázků. Jsou představené používané techniky a jsou ukázány testovací sady, které slouží k meření přesnosti vyledávacích systémů založených na podobnostním vyhledávání obrázků. Na modelech, implementovaných na základě představených technik, jsou pak prováděna různá měření, která zkoumají jejich výsledky v závislosti na vstupních datech, použitých komponentách a nastavení parametrů, přičemž speciální pozornost je věnována chování modelů pro různá rozlišení obrázků. Tyto výsledky jsou dále analyzovány a jednotlivé modely jsou vzájemně porovnány.

Klíčová slova: Vyhledávání obrázků, rozlišení obrázků, signatury, SIFT, VLAD

Title: The Impact of Image Resolution on the Precision of Content-based Retrieval

Author: Lukáš Navrátil

Department: Department of Software Engineering

Supervisor: RNDr. Jakub Lokoč, Ph.D., Department of Software Engineering

Abstract: This thesis is focused on comparing methods for similarity image retrieval. Common techniques and testing sets are introduced. The testing sets are there to measure the accuracy of the searching systems based on similarity image retrieval. Measurements are done on those models which are implemented on the basis of presented techniques. These measurements examine their results depending on the input data, used components and parameters settings, especially the impact of image resolution on the retrieval precision is examined. These results are analysed and the models are compared.

Keywords: Image retrieval, image resolution, feature signatures, SIFT, VLAD

Contents

Preface

Image Retrieval (IR) is a field of study concerned with searching, browsing and retrieving images from databases of digital images. It has been a field of study since 1970s [1]. In early years it started with methods used in a text retrieval, later in 1990s research in the Content Based Image Retrieval (CBIR) area started [2]. In the last decade we were witnesses of a massive growth in the multimedia area that led to the need of research in the IR area. With the current boom of multimedia devices a huge amount of multimedia content is produced and it is required to be able to search and explore these large collections effectively. For this purpose a lot of applications and services that are using several different approaches to the IR have been developed.

One of the possible approaches to the IR is text-based image retrieval, where text keywords are used as descriptors to index the image. These keywords are usually products of manual annotation or they can be extracted from related context, e.g. web page where image is placed. When a search is being made, user has to provide a set of keywords that is compared with keywords saved in the database. In this case only general text based retrieval methods are used, content of the image is ignored. An advantage of this approach is that general text-based search engines can be reused. The main disadvantage is the need of related relevant text description for each indexed image — there are numerous applications that need to retrieve images just from provided image content. It can be the only information we can get from users or it can be easier for users to form the query by an image than by keywords, as the familiar proverb "A picture is worth a thousand words" says.

On the contrary, in the CBIR [2, 3] features extracted from the content of the image are used for indexing. Content can refer to colours, textures or any other information derived from the image itself, not from the metadata. The query consists of an image or a set of images instead of a set of keywords like in a text-based retrieval. From given query images features or their aggregates are extracted that are compared with data extracted from indexed images and similarity is computed.

Implementations of the state-of-the-art CBIR search engines consists of several parts. For each step there are several different models or algorithms with various performance and precision in different conditions.

The thesis is focused on several state-of-the-art CBIR models and the way, how their precision is affected by quality of input data or settings of the search engines is discussed. The thesis aims on the CBIR models designed for searching in general domain of images — for specific domains (e.g. human faces or fingerprints) there are specialized models with better performance in given domain.

The results of selected models and algorithms in few well-known benchmarks used for evaluation of the CBIR engines precision are compared.

1. Introduction to Content-based Image Retrieval

Multimedia information retrieval denotes the process of retrieving data objects from databases with respect to user's needs. Content-based information retrieval [2, 3] focuses on the properties of multimedia objects. In this thesis we focus on the images — in this case properties are derived from content of the image.

For retrieving multimedia objects from database, users have to formalize their needs into a query. There are several query techniques, the most common is the query by example [4] where the query consists of an image or a set of images — then it is called a multi-query. The example can be an image uploaded by an user, selected from database or there are search engines that can retrieve images by a sketch.

1.1 Similarity Model

An essential part of each retrieval engine is a similarity model which defines how similarity between objects is computed. In this thesis we focus on distance-based similarity models that are defined as follows:

Definition 1. Distance function
Let $\mathbb{X}$ be a set. Then $\delta : \mathbb{X} \times \mathbb{X} \to \mathbb{R}^{\geq 0}$ is a distance function if it satisfies the following properties:

- reflexivity: $\forall x \in \mathbb{X} : \delta(x, x) = 0$
- non-negativity: $\forall x, y \in \mathbb{X} : \delta(x, y) \geq 0$
- symetry: $\forall x, y \in \mathbb{X} : \delta(x, y) = \delta(y, x)$

Distance functions are used to evaluate the closeness of objects from a feature space.

Definition 2. Distance-based similarity model
Let $\mathbb{I}$ be a universe of images and let $\mathbb{F}$ be a descriptor space, $f : \mathbb{I} \to \mathbb{F}$ be a descriptor extraction function and $\delta : \mathbb{F} \times \mathbb{F} \to \mathbb{R}^{\geq 0}$ be a distance function. Then distance-based similarity model $D : \mathbb{I} \times \mathbb{I} \to \mathbb{R}^{\geq 0}$ is defined for all $i_j, i_k \in \mathbb{I}$ as:

$$D(i_j, i_k) = \delta(f(i_j), f(i_k))$$

The lower is value of $D(i_j, i_k)$, the more similar images i_j and i_k are.

1.1.1 Object Representation

A feature extraction function transforms raw image data into the representation that can be used easily in a similarity model. There are various representations that can be used, in this thesis we focus on two commonly used models — feature histograms and feature signatures [5].

Feature histograms are fixed-size vectors of real numbers that can be extracted directly from the image, or they are products of a low-level feature aggregation in fixed-sized vectors. Unlike the feature histograms, the feature signatures have variable size. It allows more flexible object representation where some images can have more centroids than other images.

Definition 3. Feature signature
Let $\mathbb{F}$ be a feature space and $C = C_1, \ldots, C_n$ a clusters of features $f_1, \ldots, f_k \in \mathbb{F}$ of object o. The feature signature S^o is defined as a set of tuples from $\mathbb{F} \times \mathbb{R}^+$ as follows:

$$S^o = \{\langle c_i^o, w_i^o \rangle | i = 1, \ldots, n\}$$

where

$$c_i^o = \frac{\sum_{f \in C_i} f}{|C_i|}$$

is a centroid representative and

$$w_i^o = \frac{|C_i|}{k}$$

represent the weight of the centroid.

1.1.2 Distance Functions

For representations described above we need to be able to compute distance between all pairs of the objects. There are various distance functions for this purpose. Some common distances that are used in this thesis and in related programs are described below.

Minkowski Distances

Probably the most popular distance functions for computing distance between two vectors of real numbers are Minkowski distances (also called L_p distances).

Definition 4. L_p distance
Let x, y be two n-dimensional vectors, then L_p distance is defined as:

$$L_p(x, y) = (\sum_{i=1}^{n} |x_i y_i|^p)^{\frac{1}{p}}$$

For $p \geq 1$, the L_p distance is a metric. Most frequently used are L_1 (called as Manhattan distance) and L_2 (Euclidean distance). The complexity of L_p distance computation is $O(n)$ so it is considered as a cheap function.

Signature Quadratic Form Distance

Signature Quadratic Form Distance (SQFD) [6] is a distance measure which is a generalization of the Quadratic Form Distance [7] for feature signatures.

Definition 5. Signature Quadratic Form Distance
Given two feature signatures $S^p = \{\langle c_i^p, w_i^p \rangle | i = 1, \ldots, m\}$ and $S^q = \{\langle c_i^q, w_i^q \rangle | i = 1, \ldots, n\}$ and similarity function $f_s : \mathbb{F} \times \mathbb{F} \to \mathbb{R}$ over a feature space $\mathbb{F}$. The Signature Quadratic Form Distance $SQFD_{f_s}$ between signatures S_p and S_q is defined as:

$$SQFD_{f_s}(S^p, S^q) = \sqrt{(w_p| - w_q) \cdot A_{f_s} \cdot (w_p| - w_q)^T}$$

where $A_{f_s} \in \mathbb{R}^{(m+n)\times(m+n)}$ is the matrix arising from applying the similarity function f_s to the corresponding centroid representatives: $a_{ij} = f_s(c_i, c_j)$. Furthermore, $w_p = (w_1^p, \ldots, w_m^p)$ and $w_q = (w_1^q, \ldots, w_n^q)$ are weight vectors of signatures S^p and S^q, respectively. $(w_p| - w_q)$ denotes concatenation of vectors w_p and $-w_q$.

The time complexity of SQFD is $\mathcal{O}((m+n)^2 \cdot \mathcal{O}(f_s))$ where m and n denote the size of signatures S^p and S^q, respectively, and $\mathcal{O}(f_s)$ denotes the complexity of the similarity function f_s. There are three typical similarity functions used in SQFD [6].

Definition 6. Minus, Heuristic and Gaussian function
Given a distance function $\delta : \mathbb{F} \times \mathbb{F} \to \mathbb{R}^+$ and a parameter $\alpha \in \mathbb{R}$, similarity functions are defined as follows:

- Minus function: $f_-(c_i, c_j) = -d(c_i, c_j)$
- Gaussian function: $f_g(c_i, c_j) = e^{-\alpha \cdot d^2(c_i, c_j)}$
- Heuristic function: $f_h(c_i, c_j) = \frac{1}{\alpha + d(c_i, c_j)}$

The minus function is parameterless, for other functions a parameter α must be specified. It is usually obtained empirically and the optimal value may depend on the dataset. The value of α is also important for indexability [8]. d is called the ground distance function, usually some cheap distance such as L_2 is used.

Perceptually Modified Hausdorff Distance

Another popular distance measure for feature signatures is Perceptually Modified Hausdorff Distance (PMHD) [9] which is based on the Hausdorff distance [10].

Definition 7. Perceptually Modified Hausdorff Distance
Given two feature signatures $S^p = \{\langle c_i^p, w_i^p \rangle | i = 1, \ldots, m\}$ and $S^q = \{\langle c_i^q, w_i^q \rangle | i = 1, \ldots, n\}$, a Perceptually Modified Hausdorff Distance $PMHD$ between signatures S^p and S^q is defined as

$$PMHD(S^p, S^q) = Max\{d_H(S^p, S^q), d_H(S^q, S^p)\}$$

where $d_H(S^p, S^q)$ denotes directed Hausdorff distance between signatures S^p and S^q that is defined as

$$d_H(S^p, S^q) = \frac{\sum_i (w_i^p \times \min_j \frac{d(s_i^p, s_j^q)}{min(w_i^p, w_j^q)})}{\sum_i w_i^p}$$

and d is a ground distance function.

Like in the SQFD, d denotes the ground distance function (usually L_2). Also weighted L_2 distance can be used when we want to increase or decrease weights of some components in the feature signatures. Optimal weighting coefficients depend on the feature extraction used and also on the dataset, impact of these values on the precision is discussed in chapter 4, further in this thesis.

1.2 Feature Extraction

As aforementioned, one of the important parts is a feature extraction. There are numerous algorithms for this task. In 1999 Lowe introduced Scale Invariant Feature Transform (SIFT) algorithm [11] which is still one of the most efficient feature extraction algorithms. SIFT, Speeded Up Robust Features (SURF) [12] and features based on the position, colour and texture [13] are used in our benchmarks.

There are two types of features — local and global features. The global features describe content of the whole image, an example of such feature is a colour histogram or MPEG-7 features [14]. On the other hand local features usually detect and describe interesting points or its neighbourhood in an image. Within one image many features are detected and the number of features can differ in each image.

1.2.1 Scale Invariant Feature Transform

SIFT is an algorithm for local feature extraction widely used in computer vision. In the first step key points are detected — they are defined as maxima and minima of the result of difference of Gaussians function applied to a series of blurred and resampled images. These keypoints are filtered, low contrast and edge response points are removed. And finally for each keypoint dominant orientation is assigned and a descriptor vector is computed as a set of orientation histograms from keypoint neighbourhood. Descriptor is a 128-dimensional real vector that is L_2-normalized. This vector is invariant to scaling, rotation and partially invariant to illumination changes and affine distortion.

There also exist algorithms derived from the SIFT (e.g. CSIFT [15], PCA-SIFT [16] or ASIFT [17]) that perform better in certain situations [18] but in general the results are comparable.

1.2.2 Speeded Up Robust Features

SURF [12] is a local feature detector partly inspired by the SIFT, that is several times faster than the SIFT with similar robustness to image transformations. It uses integral images approximation in keypoints detection and a descriptor vector is based on the sums of intensities. Descriptor vector is also L_2-normalized and it has only 64-dimensions.

1.2.3 PCT Features

Position Colour Texture (PCT) features [13] are low, 7-dimensional, image features that use feature space $\mathbb{F}$ defined as $(x, y, L, a, b, c, e) \in \mathbb{F} \subset \mathbb{R}^7$, where (x, y)

are relative coordinates of the sampled point in the image, (L, a, b) represent the colour of the sampled point in the CIE Lab colour space [19] and (c, e) values represent contrast and entropy computed from the neighbourhood of the sampling point. Each dimension is normalized into $[0, 1]$ interval.

These features are usually extracted from sampling points that are chosen randomly with using uniform or normal distribution.

1.3 Aggregating Local Features

Described feature extractors produce many individual local features for each image. To compute the distance between two images, we can use directly these features or we can aggregate them in some compact representation and then compute distance between these aggregates.

We distinguish two types of aggregation — a global and a local aggregation. In the first case we need a global vocabulary which is then used in the clustering of features of each image. On the other hand, in local aggregation the clustering of features is made separately for each image without any global vocabulary.

An example of global aggregation are models based on Bag of Words (BoW) model, aggregation of features into feature signatures is the example of local aggregation.

1.3.1 Bag of Words Model

BoW (also called Bag of Features or Bag of Visual Words) [20] is a model for the CBIR based on the Vector model [21] used in the text retrieval. It aggregates various number of local descriptors from the image in one global fixed-size descriptor called Bag of Visual words and effectively solves the problem that each image can have different number of detected local features. Also, it is much faster to compare the resulting descriptors than individual features.

To create a descriptor D of size d of image I from a set of features F_I we need a codebook (also called a vocabulary) $C = c_1, \ldots, c_d$ which is a set of d vectors with the same dimensionality like the features in F_I. This codebook is usually obtained by k-Means [22] clustering on some training set of the image features. These features can be extracted from a current image database or from an independent set of images.

The BoW is a histogram of numbers of the image features assigned to each vectors $c_1, \ldots, c_d$. Thus it produces d-dimensional vector of non-negative integers that is subsequently L_1 or L_2-normalized and also the tf/idf scheme [23] can be applied and the vector is normalized again.

The distance between descriptors representing images can be computed with L_1, L_2 or cosine distance [23].

1.3.2 Vector of Locally Aggregated Descriptors

There were several models based on the BoW introduced in recent years, that outperform the classic BoW model [24, 25] both in case of accuracy and efficiency. One of the state of the art extension is the Vector of Locally Aggregated Descriptors (VLAD) [24]. The VLAD is a simplified version of Fisher Kernel [26] model,

that creates vectors with lower dimensionality and with better search accuracy than classic BoW.

Building VLAD Vectors

As for the BoW, the first step is to create a codebook $\{c_1, \ldots, c_K\}$ with K-Means clustering (L_2 distance is used). Each image is represented by a single VLAD vector that is a concatenation of residual vectors $v_1, \ldots, v_K$ where

$$v_i = \sum_{x_t : NN(x_t) = i} x_t - c_i$$

and $NN(x_t) = i$ if c_i is the closest neighbour from a codebook vector $\{c_1, \ldots, c_K\}$ to the descriptor x_t. The resulting vector is Kd dimensional where d is the dimension of the local features and K is the size of codebook. Then the vector can be L_2-normalized. Another option is to reduce dimension with Principal Component Analysis (PCA) [27], it can also lead to better accuracy [24].

Residual Normalization

An improvement to the VLAD presented by Delhumeau et al. [28] is to L_2-normalize each residual vector in $v_1, \ldots v_K$ and then to normalize the whole VLAD vector. The motivation is that norms of residual vectors may vary significantly and individual local descriptors contribute unequally to the VLAD representation.

1.3.3 Feature Signatures

Creation of the feature signatures is based on the adaptive version of K-means clustering [13]. As input we provide extracted features $f_1, \ldots, f_n$ from an image I and seed features $s_1, \ldots, s_m$ that are used as initial centroids of clusters. In each iteration of K-means, features $f_1, \ldots, f_n$ are assigned to the closest (L_2 distance is used) centroids, new centroids are computed and clusters that have centroids closer than specified threshold are merged together. Also clusters smaller than specified threshold are removed and points are reassigned in the next iteration. The clustering ends after a specified number of iterations and centroids from the last iteration are used as centroid representatives in the resulting feature signature and counts of the features in the clusters are used as the weights of centroids.

An example of the image with generated feature signatures is displayed in figure 1.1.

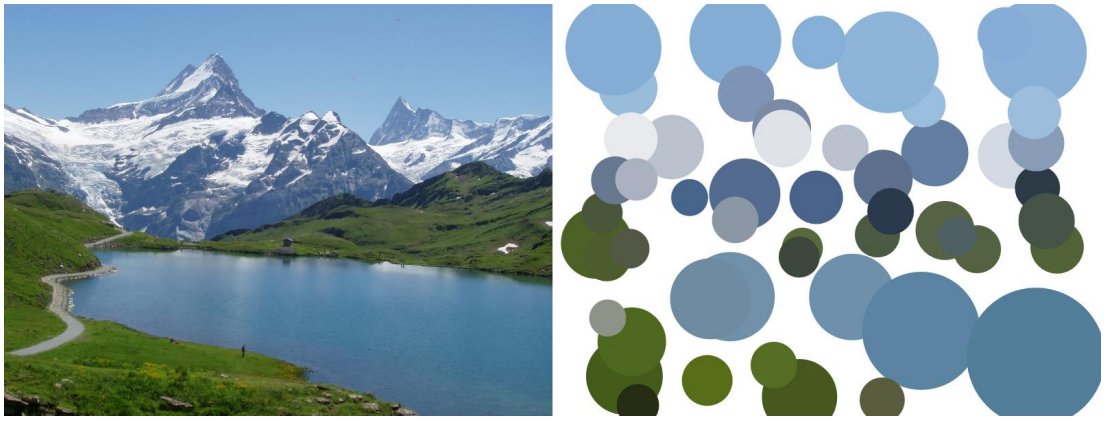


Figure 1.1: Image from Holidays dataset [29] with extracted PCT features and aggregated into the feature signatures

1.4 Similarity Queries

A query is a formalization of user's need, that is passed to the retrieval engine. Based on this query the engine should retrieve subset of database with respect to the similarity model and a given query. In this thesis the k-nearest-neighbours (k-NN) queries [30] are used that are defined as follows:

Definition 8. k-NN query
Let X be a set, $\delta : X \times X \to \mathbb{R}^{\geq 0}$ be a distance function, $q \in X$ be a query object and k be the number of wanted results. The k-nearest-neighbour query is defined as $kNN(q, \delta, X) = \{R \subseteq X, |R| = k \wedge \forall x \in R, y \in X \setminus R : \delta(q, x) \leq \delta(q, y)\}$

Hence the k^{th} nearest neighbour is the object from a set X with k^{th} smallest distance from the query object q with respect to the distance function δ. q is present in $kNN(q, \delta, X)$ if and only if it is also contained in X.

Alternative to k-NN is a range query [30] where all the objects with distance to the query object q lower than specified threshold r are returned. But k-NN is more practical in the most of real applications — we do not have to care about optimal value of r (the size of the result can vary significantly depending on the query or the dataset) and k-NN also enables paging of the results.

Definition 9. Range query
Let X be a set, $\delta : X \times X \to \mathbb{R}^{\geq 0}$ be a distance function, $q \in X$ be a query object and $r \geq 0$ be a distance. The range query is defined as $R(q, r, \delta, X) = \{x \in X, \delta(x, q) \leq r\}$

2. Benchmarks

For purpose of evaluating precision of the CBIR systems, numerous benchmarks were created. Each benchmark consists of a set of images and a ground truth. It defines which objects should be used as queries and which images are considered to be the correct results for each query. Each benchmark has its own scoring system which produces a single value that can be compared with other measurements.

2.1 Measuring the Precision

Scoring systems of most benchmarks are using the Mean Average Precision (MAP) which is based on two basic values — precision and recall [23]. Precision says the ratio of the relevant objects in some query result, the value of recall is the fraction of the objects that are relevant to the query that were retrieved.

Definition 10. Precision and Recall
Let I be a database of images and q be a image retrieval query. Furthermore Rel is a set of images that are relevant to query q and Ans is a set of images returned by retrieval engine for query q. Then P denotes precision and R denotes recall and they are defined as follows:

$$P = \frac{|Rel \cap Ans|}{|Ans|}$$

$$R = \frac{|Rel \cap Ans|}{|Rel|}$$

In the systems that return result as a ranked sequence of objects, we also want to consider the order in which the results are returned. It is done by computing precision and recall at every position in the ranked sequence. Then we can plot a precision-recall curve, plotting precision as a function of recall. We use definition of average precision that is used in evaluation protocols of Holidays [29] and Oxford Buildings [31] datasets, where average precision is defined as the area under the precision–recall curve:

Definition 11. Average precision

$$AP = \sum_{k=1}^{n} \frac{P(k-1) + P(k)}{2} (R(k) - R(k-1))$$

where n denotes the number of retrieved objects, $P(j)$ is the precision of the first j retrieved objects and $R(j)$ is the recall of the first j retrieved objects. The values for $j = 0$ are defined as follows: $P(0) = 1, R(0) = 0$

Definition 12. Mean average precision
Let Q be a set of queries and let $AP(q)$ be an average precision of the query $q \in Q$. Then mean average precision $MAP(Q)$ of set of queries Q is defined as follows:

$$MAP(Q) = \frac{\sum_{q \in Q} AP(q)}{|Q|}$$

2.2 Datasets

Datasets described in the following sections were selected as convenient benchmarks for the needs of this thesis. All of them are well-known and are frequently used in the state-of-the-art articles about the CBIR. Two of them provide only low-resolutions pictures, others provide mid-to-high-resolution images that are required for our experiments. Measured results are discussed in chapter 4 further in this thesis.

2.2.1 The Oxford Buildings Dataset

The Oxford Buildings Dataset [31] consists of 5062 images which were retrieved from Flickr by searching for Oxford landmarks. There are 11 different landmarks each represented by five query images. For each of 55 queries there are defined four groups of images:

- Bad - query object is not present
- Junk - less than 25% of the query object is visible, or there are very high levels of occlusion or distortion
- OK - more than 25% of the query object is clearly visible
- Good - nice, clear picture of the query object is visible

Dataset is available on-line [32].

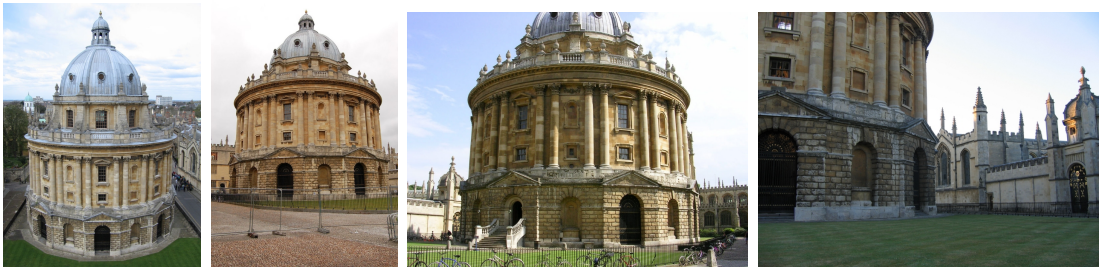


Figure 2.1: Example of images from Oxford dataset [31]. The query object on the left, then example from Good, OK and Junk categories.

Evaluation protocol

A modification of the MAP is used for computing the precision. Images from Junk group are skipped from evaluation — images from groups Good and OK are considered as relevant results otherwise images from group Bad are not relevant.

2.2.2 INRIA Holidays dataset

The Holidays dataset [29] consists of 1491 images which are divided into 500 groups. For each group one query image and $1-12$ corresponding relevant images are defined but for the majority of queries there are up to 3 corresponding images. Each group represents a distinct scene or object.

The images are in high-resolution, most of them have at least 1 million pixels, but the resolution of each image can be different — the smallest image has resolution 640×480, the largest 3888×2592.

Dataset is available on-line [33].

Figure 2.2: Example of images from Holidays dataset [29]. The query object on the left, other objects are correct retrieval results for this query.

Evaluation Protocol

A modification of the MAP is used — the query images are not counted as true positives, they are skipped (they are sorted as Junk in the Oxford dataset and the same protocol can be used).

2.2.3 UKBench Benchmark

The University of Kentucky Recognition Benchmark [34] contains 10200 images, all in resolution 640×480 pixels. Dataset consists of 2550 groups of 4 images each. All pictures in one group capture the same object from different view angles.

Dataset is available on-line [35].

Evaluation protocol

In this benchmark each image is a query object. The score for each query is then computed as a size of intersection of images retrieved by a 4-NN query and images from same group as the query image. For each query the score is between 0 (no, even not the query, images from group were retrieved) and 4 (all images from the group were retrieved). The final result is defined as average of scores of all queries.

Figure 2.3: Example of images from UKBench dataset [34].

2.2.4 Profimedia Benchmark

Profimedia benchmark is a subset of Profimedia dataset [36] that contains 21993 images, all in resolution 150×150 pixels, that are divided into 100 classes, there is

a single query image defined for each class. Some images were resized from their originals without keeping the image size ratio, hence they are slightly deformed.

Images in one group contains images that are visually similar but usually different objects are captured, not the same objects, just in different scenes like in benchmarks mentioned before. The MAP is used for measuring precision.

2.2.5 TWIC Benchmark

The Thematic Web Images Collection (TWIC) [5] is a dataset containing 11555 images divided into 200 classes. This dataset was created from results found for various keywords searched by Google image search. Images were then checked by several people and final objects were selected.

Each class contains at least 50 visually similar objects placed on a heterogeneous background and also, like in Profimedia dataset, the captured objects are usually different in most of the classes. All images have 150×150 pixels. For each class one query image is defined. The MAP is used for measuring precision.

3. Implementation

For the evaluation of all the measurements presented in this thesis, we developed a solution that is based on a framework for multimedia exploration [37, 38]. This framework was developed as a student software project on our faculty. The author of this thesis is one of the authors of the exploration framework, other members of the development team are Přemysl Čech, Tomáš Grošup, Jakub Kinšt and Miroslav Macík.

In this framework we implemented most of the models required for needs of this thesis. The author of this thesis was responsible for the implementation of the BoW based models and related infrastructure. However some general parts used in the CBIR and required by this thesis were not present in the framework, they were implemented during the work on this thesis. Also a completely new project used for benchmarks evaluation was created and integrated into the framework.

From the beginning, the framework was built with emphasis on the extensibility and the robust architecture that would allow to use the complex models composed by various combinations of sub-components easily.

In the section 3.1 we briefly describe the main architecture of the framework which was created in cooperation with Tomáš Grošup [38] and Přemysl Čech [39]. In further sections will be described components which are related to this thesis and were developed by the author of this thesis.

3.1 Framework Basics

The framework is written in the C# 5.0 language and is separated into several projects — each project is a separate assembly. The framework is built as a web server application with ASP.NET MVC and ASP.NET WebAPI frameworks. However for the purpose of this thesis just some core parts were reused since there is no graphic interface.

Main principles and components related to the subject of this thesis are described in the following paragraphs. You can find more information about the architecture and other parts of the framework in the thesis of Grošup [38], the main software architect of this framework, or in the documentation of the Exploration framework included in the attachment of this thesis.

3.1.1 Dependency Injection

To ensure easy substitution of components, the framework was built using the Dependency Injection (DI) design pattern [40]. Individual components do not create their dependencies directly, but each component requires their lower-level dependencies as constructor parameters. When the instance of the component is created, the Inversion of Control (IoC) container (Autofac library [41] in our case) creates the dependencies based on the registrations of components. This enables to interchange individual sub-components without the need to touch the code of the parent components.

The IoC does three following important things:

- Registration of components — as first, based on the configuration, individual implementations of the components, that will be used when an instance of component is required, are registered. Each component registration includes its scope which determines the life-time of the instance and defines when a new component instance is created and when existing instance is reused.
- Resolving of components — when an instance of a component is requested, the IoC container resolves the requested instance of component based on the rules from component registrations. This is done recursively from the requested component to their dependencies.
- Releasing of components — the IoC container guarantees correct releasing of the created instances.

Two types of the components registrations are used in the framework — registrations fixed in the code and registrations based on the configuration parameters. In the second case the common use case is that we have several implementations of some interface and e.g. user can select, in the GUI, which implementation would be used. This is also used in the benchmarks — e.g. the requested implementation of the feature extractor is chosen by the value taken from benchmark configuration, that is produced dynamically.

3.1.2 Caching the Results

To compute reusable results only once, some data are cached and can be re-used in the next computations. In the context of this thesis, it is extremely useful for benchmark data sources, where extraction of features from images and their aggregation into the VLAD vectors or feature signatures are time-consuming operations. Changing some of the configuration parameters does not need to recompute the whole data source, but it just influences the subsequent operations. In that case we reuse the existing data source and modify it to match the requested configuration. The examples of these parameters are e.g. values of α, λ or flag whether the residual normalization would be applied to the VLAD vectors.

Each serializable class can be persistently stored and later loaded using the class `PersistentCache`, the caller just has to build the unique key that is used for storing and retrieving the data.

3.2 Main Components

In this section we briefly describe the main components that are used in programs related to this thesis.

3.2.1 Data Sources

A data source contains data structures that store extracted data from images (e.g. feature signatures) and methods that manage the extraction of this data. The extraction is not executed by the data source itself, it just calls proper subcomponents.

In the exploration framework there are several built-in data sources, e.g. data source that downloads images from fulltext search results or data source that loads images from a personal Facebook account. For the needs of this thesis we created two data source implementations: `BenchmarkSignatureDataSource` and `BenchmarkBoWDataSource`. The first one extracts and stores the feature signatures from the provided images, the second one stores the extracted bags of visual words. The bags of visual words are generic and they can be products of aggregation into the VLAD vectors or into the standard bag-of-words representation. The data source has no impact on used aggregation method — it is determined by the registrations of sub-components of `BagOfWordsDescriptorCreator`.

In figure 3.1 you can see a basic diagram of dependencies in the data source `BenchmarkBoWDataSource`. Only components related to this thesis are displayed, trivial dependencies (e.g. dependency on `SimilarityConfiguration`) are also hidden in this diagram.

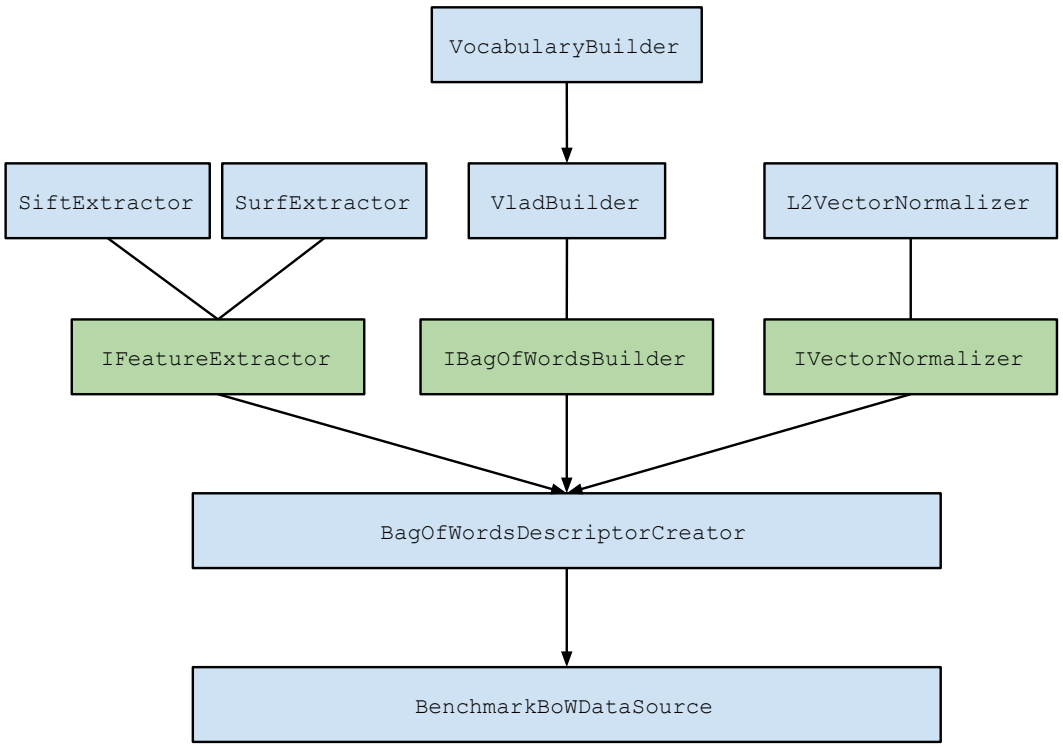


Figure 3.1: Basic diagram of dependencies in `BenchmarkBoWDataSource`. Blue items are concrete implementations, green items are interfaces. Arrows denote dependency between components, class connected with a line with an interface means that class can be registered as an interface implementation in the IoC container.

3.2.2 Feature Extractors

In this thesis three different feature extractors were used. The `SiftExtractor`, the `SurfExtractor` and the `PctSignatureExtractor`. First two extractors produce a set of extracted features for each image, unlike the `PctSignatureExtractor`

also makes the aggregation into the feature signatures that are returned as the result.

For extracting SIFT and SURF features we use extractors from EmguCV library [42] — a .NET wrapper to the OpenCV library [43].

The `PctSignatureExtractor` uses the implementation provided by the authors of feature extraction algorithm [13].

3.2.3 Bag of Words Builders

For an aggregation of the image features into the feature histograms, there are two different BoW builders implemented — `VLADBuilder` and `BagOfWordsBuilder`. Both of them aggregate a set of features into the histograms based on the implemented rules and the provided vocabulary. Existing vocabulary can be loaded from the cache or when there is no suitable one, a new vocabulary is trained.

Vocabulary Builder

The vocabulary is trained from all the features from all the images in the dataset using the K-means BoW trainer from EmguCV library [42].

3.2.4 Distance Providers

Distance providers are responsible for computing the distance between objects from given type of data source. Please note that only identifiers of the objects are given when the distance is computed, the provider takes the required data from the data source itself.

In this thesis we used these providers: the `SqfdDistanceProvider` and the `PmhdDistanceProvider` for computing the SQFD and PMHD distances for feature signatures and `L2DistanceProvider` for computing distances between histograms.

The `SqfdDistanceProvider` and the `PmhdDistanceProvider` were implemented together with other authors of the framework.

3.3 Implementations of Benchmarks

For every benchmark mentioned previously in this thesis was created a class that implements interface `IBenchmark`. It provides methods for the complete evaluation, starting with getting all images, ending with calculation of precision. The signature of the `IBenchmark` interface is following:

```
public interface IBenchmark
{
    List<StaticImage> GetAllImages();
    List<BenchmarkQuery> GetQueries();
    bool CanBeQueryImage(StaticImage image);
    double ComputeAveragePrecision(BenchmarkQuery query,
        List<StaticImage> result);
    uint kNNResultSize { get; }
}
```

Please note that images from the datasets are not included in the attachment due to the copyrights. But you can find instructions where to get the data in the benchmarks documentation in attachment.

3.3.1 Benchmark Configurations

In the framework there is one main configuration class `SimilarityConfiguration` that keeps all settings related to the similarity models that are used across the whole framework. This class e.g. includes parameters that are used in the feature extraction, vocabulary creation or in building the VLAD vectors.

For the purpose of this thesis we derived from `SimilarityConfiguration` a new child class called `BenchmarkSimilarityConfiguration`, where there are additional parameters used by benchmarks.

There are also two classes called `BenchmarkSignatureConfigurationProvider` and `BenchmarkBoWConfigurationProvider`, which provide predefined configurations that can be used in the benchmark evaluations.

3.3.2 Benchmarks Evaluation

We implemented two classes that are used for evaluating of benchmarks. These classes are called `BenchmarkBoWEvaluator` and `BenchmarkSignatureEvaluator`. You can find there methods that can be called by any other applications as well as methods that are created as NUnit test framework [44] methods. They can be run or debugged directly from the development environment using any installed NUnit tests runner.

Both evaluators are using different data sources and also there are some specific operations for each model, but the basic scenario how the benchmark is evaluated is as follows:

- Get benchmark configuration from the configuration provider.
- Select the proper size of the images.
- Create the instance of a benchmark class.
- Register components in the IoC container based on the benchmark configuration.
- Build the identifier of the data source and try to get cached data source. If the data source was not cached, build a new instance of the data source.
- Prepare the data source to match the requested benchmark configuration (e.g. set correct α or λ values)
- Optional: look for the computed distance matrix in the `PersistentCache`. If the distance matrix is found, skip the next step.
- Initialize the requested distance provider and create index that is used for executing k-NN queries.
- Using the created index or loaded distance matrix, evaluate all the benchmark queries.

- Optional: store the computed distance matrix in the `PersistentCache`.
- Evaluate the precision of all benchmark queries.
- Write the results in the file.

3.4 Other Tools

For debugging the PCT features [13] extraction and aggregation into the feature signatures we developed a small tool that can be used for visualization of extracted data. You can find it in class `SignatureVisualizer`. As well as benchmark evaluators, it can be launched as NUnit [44] test, also the basic scenario, how it works is very similar. As first it requires building the data source. When there is a cached instance, it is used, otherwise a new data source is created.

In figure 1.1 you can see the sample that was produced by this tool.

4. Results

This chapter refers to the results measured on the benchmarks introduced in chapter 2 using the implementation described in chapter 3.

Results presented in this chapter are focused on retrieval precision of models. Another important quality indicator is the efficiency of the retrieval process that is not much discussed in this thesis. Required efficiency usually varies depending on the type of application, size of the dataset and other factors. Different search engines have various performance bottlenecks and it is up to the authors to balance precision and performance requirements for a given application.

All the data presented in this chapter can be found in the spreadsheet included in the attachment. There are all charts used in this section as well as raw computed data that can be also used for further analysis.

4.1 Configuration

All results were computed using the models that were described in chapter 3. These models have a lot of configuration parameters which influence the resulting precision. It was not possible to try all the meaningful values for all parameters because of a big amount of parameters — the number of required simulations depends exponentially on the count of examined parameters and there can also be correlations among different parameters. For this reason, we selected several parameters for which we supposed that they will have the biggest impact on results. The main goal is to discuss the impact of the image size to the optimal values of these parameters.

Models used in this thesis can be divided into two categories — signature-based models and BoW-based models.

4.1.1 Signature Based Model

This model is based on PCT features described in section 1.2.3, aggregated into the feature signatures representation described in section 1.3.3 and on distance functions suitable for feature signatures (SQFD and PMHD) described in section 1.1.2.

Feature Extraction and Aggregation

The process of feature extraction and aggregation into the feature signatures has several parameters, here are the most important ones:

- Number of sampling points and initial number of clustering seeds.
- The distribution used for generating the sampling points — whether uniform or normal distribution is used, and when normal distribution is used, the value of standard deviation.
- Minimal square distance and minimal cluster weight — this is used during the K-Means clustering. Default values (0.08 and 8) were used, as a future

work it could be examined whether these values shouldn't been adjusted with increasing number of sampling points.

- Weights of the components of the signature — it is possible to give larger weight e.g. to the colour part of the feature. We kept equal weight of all components. Optimal values of the weights varies a lot depending on the dataset and also it is possible to control the weights of the feature parts during the distance computation, although weighting during the extraction can lead to even better results.

Impact of these parameters is discussed in section 4.2.1

Distance Functions

We have used two distances for the feature signatures SQFD and PMHD. The SQFD was used with Gaussian similarity function defined in section 1.1.2. This function has a parameter Alpha (α). Impact of its value is discussed in section 4.2.2 further in this thesis.

In both functions we applied L_2-weighted ground distance function, in which we assigned different values to the coefficient that weights the position component (x, y), other components had the weight equal to 1. Further in this thesis the weight of the position component is called Lambda (λ) and we assigned its value from interval $[0.1, 10]$.

4.1.2 BoW Based Model

As a representative of the BoW-based models we selected the VLAD model. It is one of the state-of-the-art extensions of the BoW model, which outperforms the BoW both in case of accuracy and efficiency and it is easy to be implemented and debugged.

Feature Extraction

We used the VLAD-based models with two well-known local feature extractors — SIFT and SURF, both were briefly described in sections 1.2.1 and 1.2.2. For both SIFT and SURF we used extractors from EmguCV library [42] with default extraction parameters recommended by the documentation [45], with unrestricted number of features per image.

VLAD Parameters

The VLAD-based models have two main parameters — the size of the VLAD vector K (number of centroids in a vocabulary — we used predefined values from interval $[1, 2048]$) and a flag if the residual normalization described in section 1.3.2 would be used. The advantage of residual normalization is that it can be applied to the VLAD vectors, created without residual normalization, just before the distance computation. It means that vocabulary creation and assignment of features to the closest centroid, two the most time-consuming operations, can be done only once per each VLAD size.

The resulting vectors are L_2-normalized and L_2-distance is used for computing distance between two vectors.

Vocabulary Creation

An important part of all the BoW-based models is creating a vocabulary. We use k-Means-based trainer implementation from EmguCV library [42] with restricted maximum number of iterations to 5.

Vocabulary is created from all extracted features from all images in a given dataset. For each dataset, the size of the VLAD vector, the size of images in the dataset and used feature extractor, its own vocabulary is created.

4.1.3 Images

For purpose of observing the impact of the image size, we created several different image size classes. Each class represents images from a given dataset that were resized from original images to a required size with respect to original aspect ratio. The size is determined by the length of the longer edge of the image where the length is in pixels and it is taken from the following set: {80, 160, 320, 480, 640, 800, 960, 1120, 1280}. There is also a class, that contains original images without any resizing, called *Original*.

If any original image is smaller than requested size, it will be kept in original size — it means that some images can be smaller than others in the same class, if they were not big enough in their original size. Further in this thesis we denote the class of images resized to maximum size of X pixels as a *Xpx images*, e.g. *800px* means that all images' size is less than or equal to 800 pixels. Some datasets do not provide high-resolution pictures, for these benchmarks we use just a meaningful subset of available image size classes.

4.2 Feature Signatures Approach

In this section we will discuss performance of a model based on the PCT features aggregated into the feature signatures.

4.2.1 Feature Extraction and Aggregation

At first we need to extract image features and build feature signatures. These steps are very important — when the feature signatures are extracted wrong, it influences all the following steps.

Number of Sampling Points

One of the parameter that influences the extracted feature signatures is the number of sampling points and clustering seeds. Authors of the signature extractor [13] used the extractor mainly for small images (width and height under 150 pixels), we tried to use it also for high-resolution images. For small images there is empirically obtained optimal value of 2000 sampling points and 500 seeds. With increasing number of sampling points and seeds (we kept the same ratio, so for 4000 sampling points we used 1000 seeds etc.) the signatures can have larger amount of centroids, but it also depends on the complexity of the image.

In figure 4.1 there is an example of less complex image with extracted features. The number of detected centroids did not increase a lot with increasing number

of sampling points, however in figure 4.2 we can see that the number of centroids is increasing a lot with increasing number of sampling points.

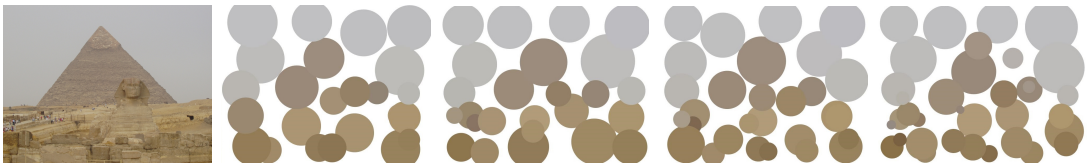


Figure 4.1: Original less structured image with the width of 800px from the Holidays dataset [29] and feature signatures extracted from 2000, 4000, 8000 and 16000 sampling points.

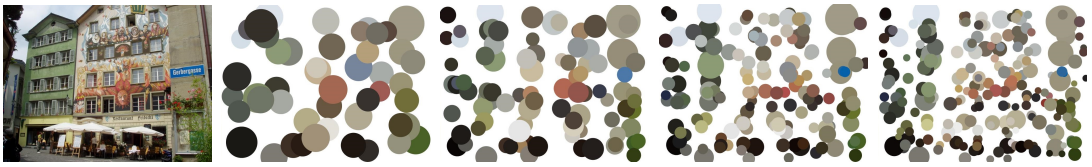


Figure 4.2: Original complex image with the width of 800px from the Holidays dataset [29] and feature signatures extracted from 2000, 4000, 8000 and 16000 sampling points.

However a larger number of centroids does not result in a better accuracy as shown in figure 4.3. More than 4000 sampling points lead to worse results both for 160px and 800px images. For 160px images increasing the number of sampling points from 2000 to 4000 brings a minor improvement, but for 800px images the growth is more significant.

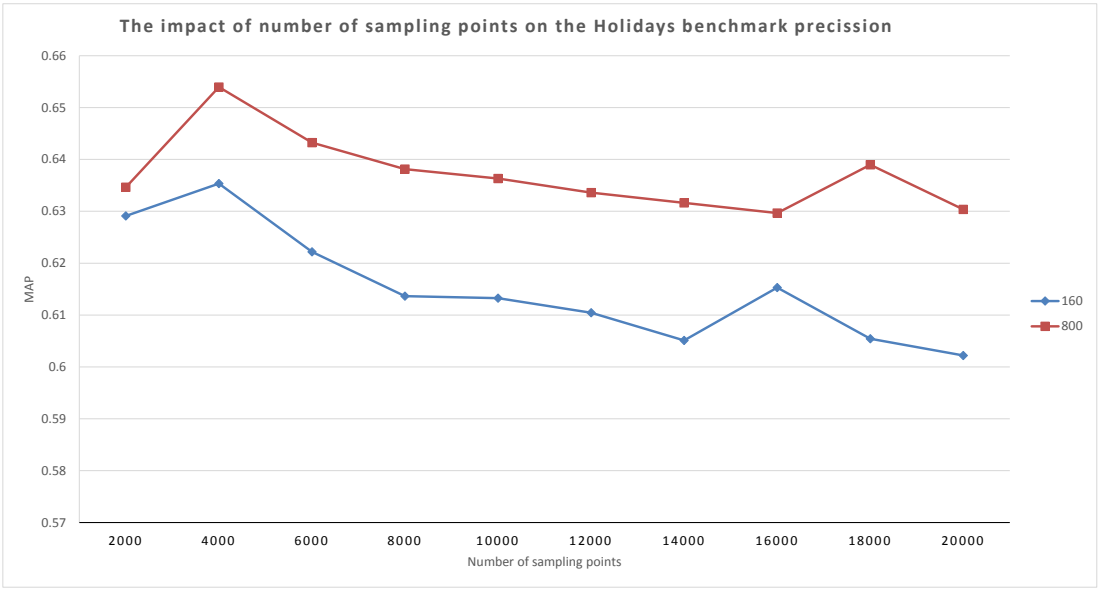


Figure 4.3: Precision for various number of sampling points for 160px and 800px images from the Holidays dataset, PMHD distance and $\lambda = 0.1$.

Sampling Distribution

Some benchmarks (e.g. TWIC) are specific because the searched objects are always placed in the middle of the image. One possible approach how to get a better precision can be using normal instead of uniform distribution when generating the sampling points and seeds. In figure 4.4 we can see the original image from the Holidays benchmark and extracted feature signatures with uniform and normal distribution.

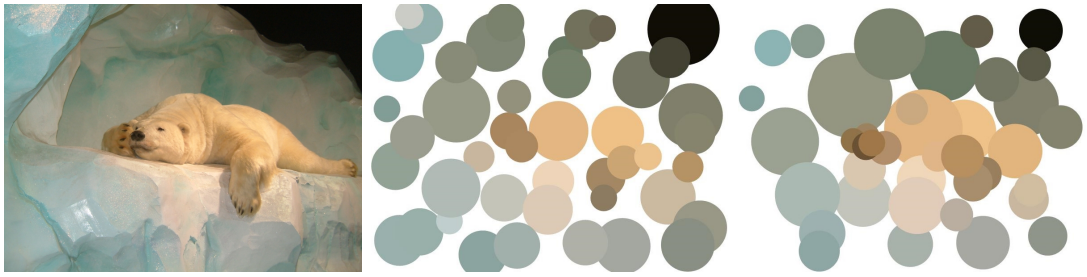


Figure 4.4: Original 800px image from the Holidays dataset [29] on the left, feature signatures extracted with uniform distribution in the middle and feature signatures extracted with normal distribution with $\sigma = 0.3$ on the right.

In figure 4.5 we can see the values of the MAP for various benchmarks. Except the last column, the sampling points were generated from normal distribution with mean equal to 0.5 and standard deviation equal to σ. When the randomly generated value was out of required interval $[0, 1]$, a new value was generated.

Only in the TWIC benchmark the usage of normal distribution brings a significant improvement, but in other benchmarks when normal distribution with $\sigma = 0.3$ is used, the resulting MAP is approximately the same as when uniform distribution is used. It seems that normal distribution with $\sigma = 0.3$ is the optimal value — in no benchmark it decreases the MAP significantly but for some datasets it can improve the precision a lot.

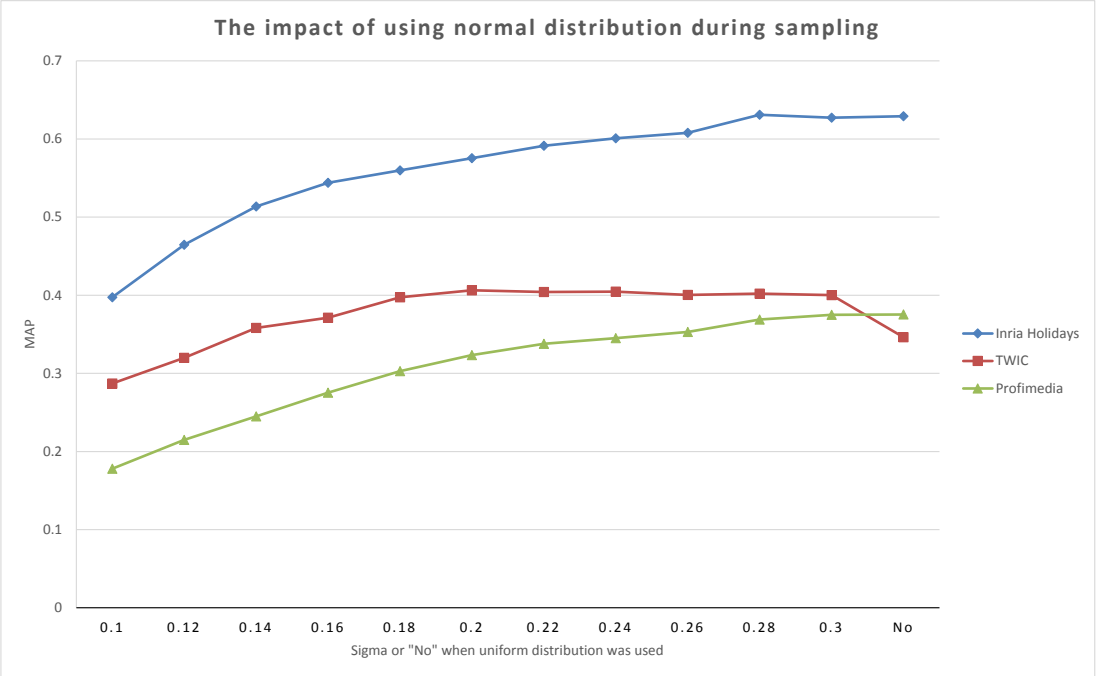


Figure 4.5: The MAP for various benchmarks with 160px images and various values of σ with normal distribution compared to uniform distribution in the last column.

4.2.2 Distance Functions

In both SQFD and PMHD distance functions we use the parameter λ. In this section we show that the correct value of λ parameter can lead to a better precision. In SQFD distance we have one additional parameter α, its value on precision is also discussed in this section.

Weighting Centroid Distances

The lower is the value of λ the lower is the weight of position, when the distance between two centroids is computed. For $\lambda > 1$ the weight of position is more important than the colour, contrast and entropy, for $\lambda < 1$ the weight of position is lower than the weight of other parts.

It was expected that the optimal value of λ would depend on the dataset, but it is interesting that it also depends on the distance that is used. As we can see in figure 4.6, in Holidays dataset the lower is the value of λ the higher is the MAP when the PMHD distance is used, but when SQFD distance is used, it leads to worse result than for neutral value $\lambda = 1$. In the Profimedia dataset the difference is even bigger, the progress of the MAP for various λ is completely opposite for the PMHD and SQFD distances. In the TWIC benchmark the progress is quite similar, but the difference of MAPs for lower values of λ is significant.

Please note that the SQFD distance has another parameter α which we did not fix in figure 4.6, but the best result for all possible α values was taken. Nevertheless when we fix the value of α to an average value, the values change a little bit, but the trends between PMHD and SQFD for all benchmarks remain the same.

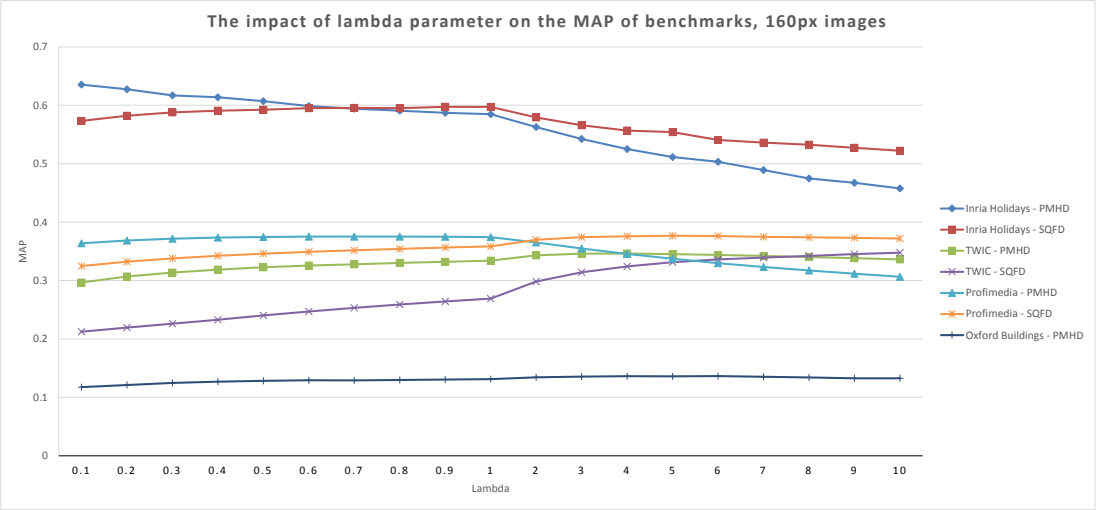


Figure 4.6: MAP for various benchmarks, distances and values of λ. 160px images and 2000 sampling points with uniform distribution were used.

An Optimal Value of Alpha for SQFD

The SQFD distance with the Gaussian function requires a value for parameter α. An advantage of this parameter existence is that we can control the indexability, a disadvantage is that some values can decrease the precision. As it is shown in figure 4.7, the optimal value depends a lot on the dataset — a larger α in the Holidays benchmark leads to a better precision, but in the TWIC and Profimedia the optimal value of α is much smaller.

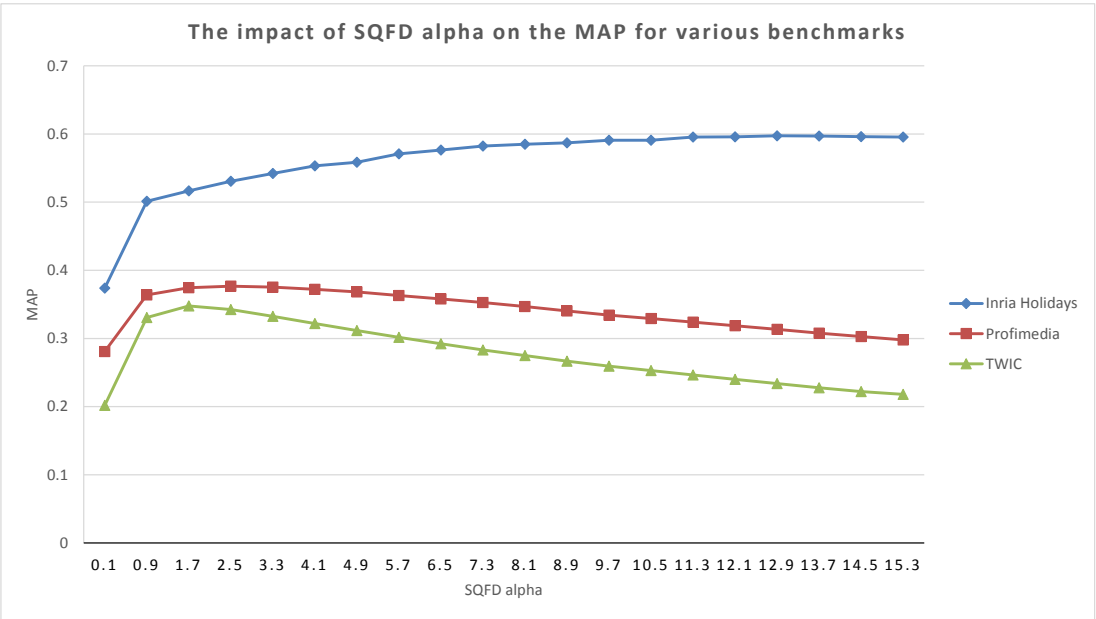


Figure 4.7: The MAP for various benchmarks with 160px images, SQFD distance and uniform sampling.

Correlation between Alpha and Lambda

In the previous two sections the optimal values of α and λ were discussed separately. Here we show that there is a correlation between values α and λ.

In figures 4.8 and 4.9 we can see the values of the MAP depending on the values α and λ. For both benchmarks we can see that the best results were obtained for low values of α and higher values of λ.

On the contrary, in figure 4.10 we can see that the optimal values for the Holidays dataset are completely opposite — we get the highest MAP for high α values and intermediate λ values.

	TWIC																			
λ \ α	**0.1**	**0.9**	**1.7**	**2.5**	**3.3**	**4.1**	**4.9**	**5.7**	**6.5**	**7.3**	**8.1**	**8.9**	**9.7**	**10.5**	**11.3**	**12.1**	**12.9**	**13.7**	**14.5**	**15.3**
0.1	0.16	0.20	0.21	0.21	0.21	0.21	0.20	0.20	0.20	0.19	0.19	0.19	0.19	0.18	0.18	0.18	0.18	0.17	0.17	0.17
0.2	0.16	0.21	0.22	0.22	0.22	0.22	0.21	0.21	0.21	0.21	0.20	0.20	0.20	0.20	0.20	0.19	0.19	0.19	0.19	0.19
0.3	0.16	0.21	0.22	0.23	0.23	0.22	0.22	0.22	0.22	0.22	0.22	0.21	0.21	0.21	0.21	0.21	0.20	0.20	0.20	0.20
0.4	0.16	0.22	0.23	0.23	0.23	0.23	0.23	0.23	0.23	0.23	0.23	0.22	0.22	0.22	0.22	0.21	0.21	0.21	0.21	0.20
0.5	0.16	0.22	0.23	0.24	0.24	0.24	0.24	0.24	0.24	0.24	0.23	0.23	0.23	0.23	0.22	0.22	0.22	0.22	0.21	0.21
0.6	0.17	0.22	0.24	0.24	0.25	0.25	0.25	0.25	0.24	0.24	0.24	0.24	0.24	0.23	0.23	0.23	0.22	0.22	0.22	0.21
0.7	0.17	0.22	0.24	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.24	0.24	0.24	0.23	0.23	0.23	0.22	0.22	0.21
0.8	0.17	0.23	0.25	0.25	0.26	0.26	0.26	0.26	0.26	0.25	0.25	0.25	0.24	0.24	0.24	0.23	0.23	0.22	0.22	0.22
0.9	0.17	0.23	0.25	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.25	0.25	0.25	0.24	0.24	0.23	0.23	0.23	0.22	0.22
1	0.17	0.23	0.25	0.26	0.27	0.27	0.27	0.27	0.26	0.26	0.26	0.25	0.25	0.24	0.24	0.24	0.23	0.23	0.22	0.22
2	0.17	0.25	0.28	0.29	0.30	0.30	0.29	0.29	0.28	0.28	0.27	0.27	0.26	0.25	0.25	0.24	0.23	0.23	0.22	0.22
3	0.18	0.27	0.30	0.31	0.31	0.31	0.31	0.30	0.29	0.28	0.27	0.27	0.26	0.25	0.24	0.23	0.23	0.22	0.21	0.20
4	0.18	0.29	0.32	0.32	0.32	0.32	0.31	0.30	0.29	0.28	0.27	0.26	0.25	0.24	0.23	0.22	0.21	0.20	0.19	0.18
5	0.19	0.30	0.33	0.33	0.33	0.32	0.31	0.30	0.29	0.28	0.27	0.26	0.24	0.23	0.22	0.21	0.20	0.19	0.18	0.17
6	0.19	0.31	0.33	0.34	0.33	0.32	0.31	0.30	0.29	0.27	0.26	0.25	0.23	0.22	0.21	0.19	0.18	0.17	0.16	0.15
7	0.19	0.31	0.34	0.34	0.33	0.32	0.31	0.30	0.28	0.27	0.25	0.23	0.22	0.21	0.19	0.18	0.17	0.16	0.15	0.14
8	0.20	0.32	0.34	0.34	0.33	0.32	0.31	0.29	0.27	0.26	0.24	0.22	0.21	0.19	0.18	0.17	0.16	0.15	0.14	0.13
9	0.20	0.33	0.35	0.34	0.33	0.32	0.30	0.28	0.27	0.25	0.23	0.21	0.20	0.18	0.17	0.16	0.15	0.14	0.13	0.12
10	0.20	0.33	0.35	0.34	0.33	0.31	0.30	0.28	0.26	0.24	0.22	0.20	0.19	0.17	0.16	0.15	0.14	0.13	0.12	0.12

Figure 4.8: MAPs for combinations of α and λ on the TWIC benchmark, 160px images, 2000 sampling points, uniform sampling

Profimedia																				
λ \ α	0.1	0.9	1.7	2.5	3.3	4.1	4.9	5.7	6.5	7.3	8.1	8.9	9.7	10.5	11.3	12.1	12.9	13.7	14.5	15.3
0.1	0.25	0.30	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.31	0.31	0.31	0.30	0.30	0.29	0.29	0.29	0.28	0.28
0.2	0.25	0.30	0.32	0.33	0.33	0.33	0.33	0.33	0.33	0.32	0.32	0.32	0.31	0.31	0.31	0.30	0.30	0.30	0.29	0.29
0.3	0.25	0.31	0.33	0.33	0.34	0.34	0.34	0.34	0.33	0.33	0.33	0.32	0.32	0.32	0.31	0.31	0.30	0.30	0.30	0.29
0.4	0.25	0.31	0.33	0.34	0.34	0.34	0.34	0.34	0.34	0.33	0.33	0.33	0.32	0.32	0.32	0.31	0.31	0.30	0.30	0.30
0.5	0.25	0.31	0.33	0.34	0.35	0.35	0.35	0.34	0.34	0.34	0.33	0.33	0.33	0.32	0.32	0.31	0.31	0.31	0.30	0.30
0.6	0.25	0.32	0.34	0.34	0.35	0.35	0.35	0.35	0.34	0.34	0.34	0.33	0.33	0.32	0.32	0.32	0.31	0.31	0.30	0.30
0.7	0.25	0.32	0.34	0.35	0.35	0.35	0.35	0.35	0.35	0.34	0.34	0.34	0.33	0.33	0.32	0.32	0.31	0.31	0.30	0.30
0.8	0.25	0.32	0.34	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.34	0.34	0.33	0.33	0.32	0.32	0.31	0.31	0.30	0.30
0.9	0.25	0.32	0.34	0.35	0.36	0.36	0.36	0.35	0.35	0.35	0.34	0.34	0.33	0.33	0.32	0.32	0.31	0.31	0.30	0.30
1	0.25	0.32	0.34	0.35	0.36	0.36	0.36	0.36	0.35	0.35	0.34	0.34	0.33	0.33	0.32	0.32	0.31	0.31	0.30	0.30
2	0.26	0.33	0.36	0.37	0.37	0.37	0.37	0.36	0.36	0.35	0.35	0.34	0.33	0.33	0.32	0.31	0.31	0.30	0.29	0.28
3	0.26	0.34	0.37	0.37	0.37	0.37	0.37	0.36	0.36	0.35	0.34	0.33	0.33	0.32	0.31	0.30	0.29	0.28	0.27	0.27
4	0.27	0.35	0.37	0.38	0.38	0.37	0.37	0.36	0.35	0.34	0.33	0.33	0.32	0.31	0.30	0.29	0.28	0.27	0.26	0.25
5	0.27	0.35	0.37	0.38	0.37	0.37	0.36	0.36	0.35	0.34	0.33	0.32	0.30	0.29	0.28	0.27	0.26	0.25	0.24	0.23
6	0.27	0.36	0.37	0.38	0.37	0.37	0.36	0.35	0.34	0.33	0.32	0.30	0.29	0.28	0.27	0.26	0.24	0.23	0.22	0.21
7	0.27	0.36	0.37	0.37	0.37	0.36	0.35	0.34	0.33	0.32	0.31	0.29	0.28	0.27	0.25	0.24	0.23	0.21	0.20	0.19
8	0.28	0.36	0.37	0.37	0.37	0.36	0.35	0.34	0.32	0.31	0.30	0.28	0.27	0.25	0.24	0.23	0.21	0.20	0.19	0.17
9	0.28	0.36	0.37	0.37	0.36	0.35	0.34	0.33	0.31	0.30	0.29	0.27	0.26	0.24	0.23	0.21	0.20	0.19	0.17	0.16
10	0.28	0.36	0.37	0.37	0.36	0.35	0.33	0.32	0.31	0.29	0.28	0.26	0.24	0.23	0.21	0.20	0.19	0.17	0.16	0.15

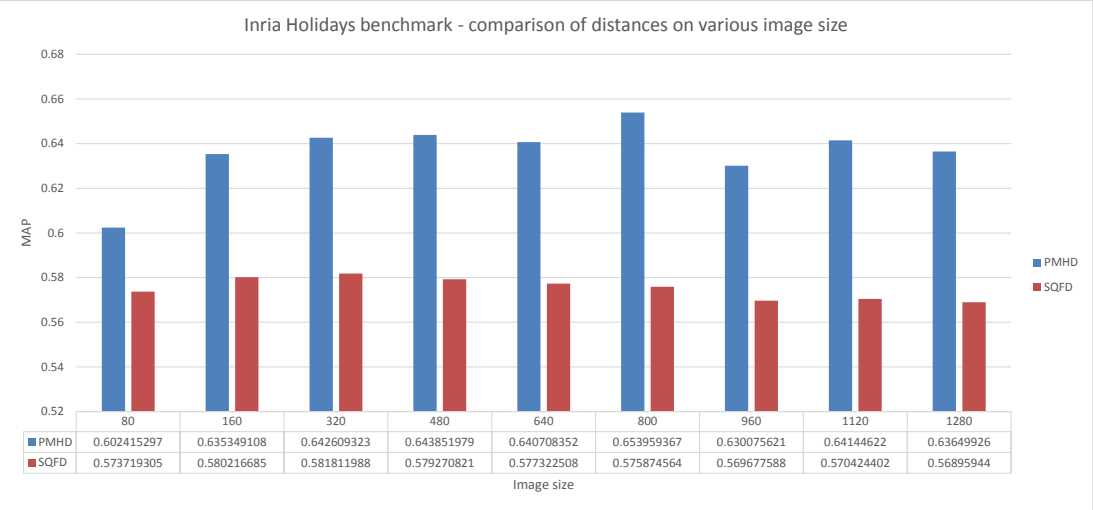
Figure 4.9: MAPs for combinations of α and λ on the Profimedia benchmark, 160px images, 2000 sampling points, uniform sampling

Holidays																				
λ \ α	0.1	0.9	1.7	2.5	3.3	4.1	4.9	5.7	6.5	7.3	8.1	8.9	9.7	10.5	11.3	12.1	12.9	13.7	14.5	15.3
0.1	0.27	0.38	0.43	0.47	0.49	0.50	0.52	0.52	0.53	0.54	0.55	0.55	0.56	0.56	0.56	0.57	0.57	0.57	0.57	0.57
0.2	0.27	0.39	0.45	0.49	0.50	0.52	0.53	0.54	0.55	0.55	0.56	0.56	0.56	0.57	0.57	0.57	0.58	0.58	0.58	0.58
0.3	0.27	0.40	0.46	0.49	0.51	0.53	0.54	0.55	0.55	0.56	0.56	0.57	0.57	0.57	0.57	0.58	0.58	0.58	0.59	0.59
0.4	0.27	0.41	0.46	0.50	0.52	0.53	0.54	0.55	0.56	0.57	0.57	0.57	0.57	0.58	0.58	0.59	0.59	0.59	0.59	0.59
0.5	0.28	0.41	0.47	0.50	0.52	0.53	0.54	0.56	0.56	0.57	0.57	0.58	0.58	0.58	0.59	0.59	0.59	0.59	0.59	0.59
0.6	0.28	0.42	0.48	0.50	0.52	0.54	0.55	0.56	0.56	0.57	0.58	0.58	0.59	0.59	0.59	0.59	0.60	0.59	0.59	0.59
0.7	0.28	0.43	0.48	0.51	0.53	0.54	0.55	0.56	0.57	0.57	0.58	0.59	0.59	0.59	0.60	0.59	0.59	0.59	0.59	0.59
0.8	0.28	0.43	0.48	0.51	0.53	0.54	0.55	0.56	0.57	0.58	0.58	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59
0.9	0.29	0.43	0.49	0.51	0.53	0.55	0.56	0.56	0.57	0.58	0.58	0.59	0.59	0.59	0.59	0.60	0.60	0.60	0.60	0.60
1	0.29	0.44	0.49	0.51	0.53	0.55	0.56	0.57	0.58	0.58	0.58	0.59	0.59	0.59	0.59	0.60	0.60	0.60	0.59	0.59
2	0.30	0.46	0.50	0.53	0.54	0.55	0.56	0.57	0.57	0.57	0.58	0.57	0.58	0.58	0.58	0.58	0.58	0.58	0.57	0.57
3	0.32	0.47	0.51	0.53	0.54	0.55	0.55	0.56	0.56	0.56	0.57	0.57	0.56	0.56	0.56	0.56	0.56	0.56	0.56	0.55
4	0.33	0.48	0.51	0.53	0.54	0.54	0.55	0.55	0.55	0.55	0.55	0.56	0.55	0.55	0.55	0.55	0.55	0.54	0.54	0.54
5	0.34	0.48	0.52	0.53	0.53	0.54	0.54	0.54	0.55	0.55	0.55	0.55	0.55	0.54	0.54	0.54	0.53	0.53	0.53	0.52
6	0.35	0.49	0.52	0.53	0.53	0.53	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.53	0.53	0.53	0.52	0.52	0.51	0.51
7	0.35	0.49	0.52	0.53	0.53	0.53	0.53	0.54	0.53	0.53	0.53	0.54	0.53	0.52	0.52	0.51	0.50	0.50	0.49	0.48
8	0.36	0.50	0.52	0.52	0.53	0.53	0.53	0.53	0.53	0.53	0.52	0.52	0.52	0.51	0.50	0.50	0.49	0.48	0.47	0.46
9	0.37	0.50	0.52	0.52	0.52	0.53	0.53	0.52	0.52	0.52	0.52	0.51	0.51	0.50	0.49	0.48	0.47	0.46	0.44	0.43
10	0.37	0.50	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.51	0.51	0.50	0.49	0.48	0.47	0.46	0.44	0.44	0.42	0.40

Figure 4.10: MAPs for combinations of α and λ on the Holidays benchmark, 160px images, 2000 sampling points, uniform sampling

4.2.3 Feature Signatures and Image Size

As we can see in figure 4.11 the results differ very slightly depending on the image size. When the PMHD distance is used, we can see a little improvement between 80px and 160px images, but then the results are very stable.

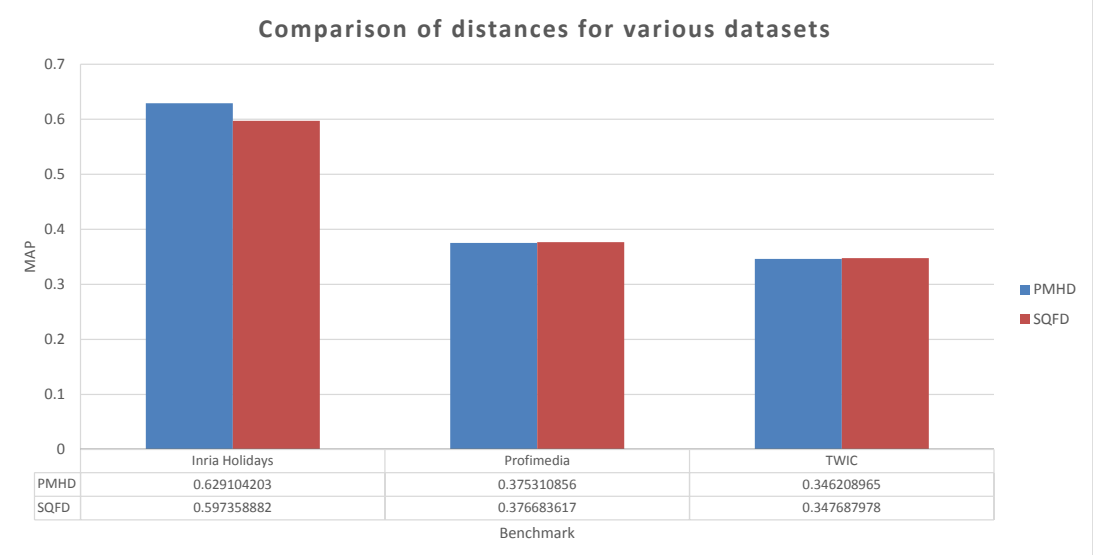


	80	160	320	480	640	800	960	1120	1280
PMHD	0.602415297	0.635349108	0.642609323	0.643851979	0.640708352	0.653959367	0.630075621	0.64144622	0.63649926
SQFD	0.573719305	0.580216685	0.581811988	0.579270821	0.577322508	0.575874564	0.569677588	0.570424402	0.56895944

Figure 4.11: MAPs in the Holidays benchmark for various image sizes. Signatures were created from 4000 sampling points

SQFD vs. PMHD

In the previous figure we could see that the PMHD distance achieves better results in the Holidays benchmark for all image sizes. In figure 4.12 we can see a comparison of distances for different datasets. The results are very similar in all datasets.



	Inria Holidays	Profimedia	TWIC
PMHD	0.629104203	0.375310856	0.346208965
SQFD	0.597358882	0.376683617	0.347687978

Figure 4.12: MAPs in various datasets for 160px images, 2000 sampling points with uniform sampling.

4.2.4 Dataset Specifics

In measurements of the signature-based models we encountered a few abnormalities that are related to particular datasets.

Holidays Optimal Values

As was shown in the section 4.2.2, some optimal values of parameters for the Holidays dataset differed significantly from the optimal values for the TWIC or Profimedia benchmarks. When we look at the Holidays dataset, we can find a few differences from the TWIC and Profimedia datasets, for example:

- Both the TWIC and Profimedia datasets contain more than 10000 images whereas the Holidays dataset consists of 1491 images only.
- In the Holidays benchmark there are 500 queries, where for each query there are only a few relevant images. The TWIC and Profimedia contain only 200 respectively 100 queries, but there are more than 100 relevant images for each query. The goal in the Holidays benchmark is to find a few specific images whereas the goal of other benchmarks is rather the image classification.
- When we compare the images from all benchmarks, we can see that in the TWIC and Profimedia datasets the images usually contain the main object clearly visible in the middle of the image. On the other hand the images in the Holidays dataset often do not have a main object. The related images often contain only a part of the query image scene or the composition of the images is quite different — the positions of the objects in the query image are different than similar objects in the related images. This is probably the reason, why we got the best results for lower values of λ, which decrease the weight of the position.
- All images in the TWIC and Profimedia datasets have the same aspect ratio — all images are squares — while images in the Holidays dataset have different aspect ratios and some images are taken as portraits, others as landscapes. Even for a portrait format query image the relevant images can be in the landscape format, and vice versa. Since the x and y positions in the features are normalized into $(0, 1)$ interval without keeping the orientation of the image, it might be another reason why a decreased weight of position part of the feature can lead to better results.

Bad Results on the Oxford Dataset

We did not mention the results of the Oxford dataset in this section. The reason is that the signature-based models have very low precision in this benchmark as can be seen in figure 4.13. It will be shown in section 4.4 that the VLAD-based models with SIFT descriptors have significantly better results in this benchmark, even for very small images.

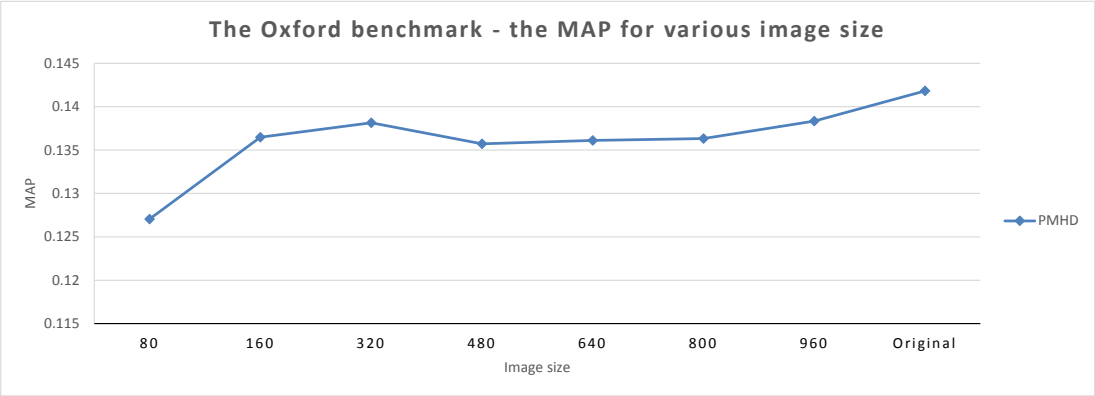


Figure 4.13: MAPs for all image sizes from the Oxford dataset, 4000 sampling points with uniform sampling.

4.3 VLAD Results

In the following paragraphs we will discuss the results of the VLAD-based models. While signature-based models perform well on small images and with increasing image size, there is no significant growth of precision, the VLAD-based models with the local features (SIFT/SURF) have optimal results when they are used with high-resolution images.

4.3.1 Feature Extractor

First we focused on the comparison of the feature extractors. As can be seen in figure 4.14, in all benchmarks that provide high-resolution images, the SIFT significantly outperforms the SURF.

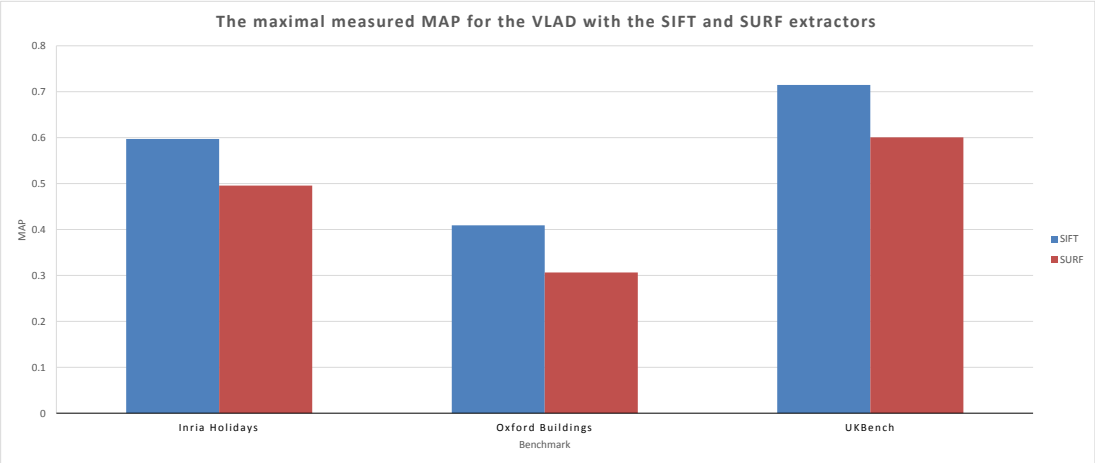


Figure 4.14: Comparison of the SIFT and SURF extractor for various datasets

We also measured precision on different image sizes, but for all image sizes the results are the same — the SIFT outperforms the SURF. For lower image resolution the difference is smaller, with a higher resolution the gap is getting

bigger. In figure 4.15 there are measured values in the Holidays dataset, in other benchmarks the results are similar.

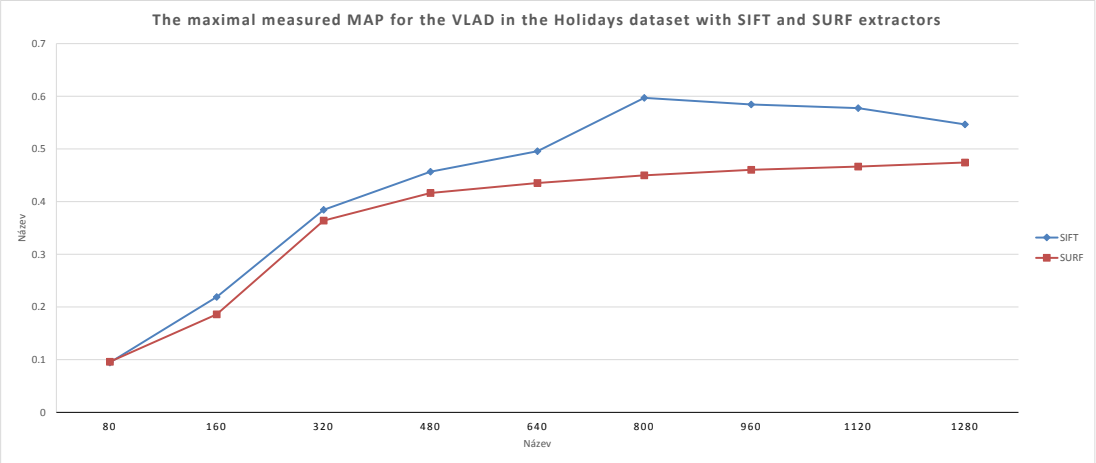


Figure 4.15: Comparison of SIFT and SURF extractor in the Holidays benchmark for various image sizes

As demonstrated in this section, the SIFT outperforms the SURF descriptors significantly, for that reason, further in this thesis, if it is not mentioned explicitly, all models are using SIFT descriptors.

The opposite results to ours — VLAD-based model with SURF has better performance than VLAD-based model with SIFT — are presented by Spyromitros-Xioufis et al. [46]. They are using different SURF feature extractor implementation from BoofCV library [47] and it can be the reason of different results. The performance of various implementations of SURF algorithm can differ, as shown by Abeles and Peter [48].

4.3.2 Image Size

The size of the images in the dataset has probably the largest impact on the precision of the VLAD-based models used with SIFT descriptors. In figure 4.16 we can see maximal reached MAPs in various benchmarks for different image sizes. In every dataset, we can see the same pattern — with an increasing image size the MAP is also growing until a certain image size. Then, for larger images the MAP is stagnating or it goes slightly down. But the breakpoint is different for each dataset — in the Oxford benchmark the precision stops growing for images larger than 320 pixels, on the contrary in the Holidays benchmark we can see a massive growth until it reaches 800 pixel images.

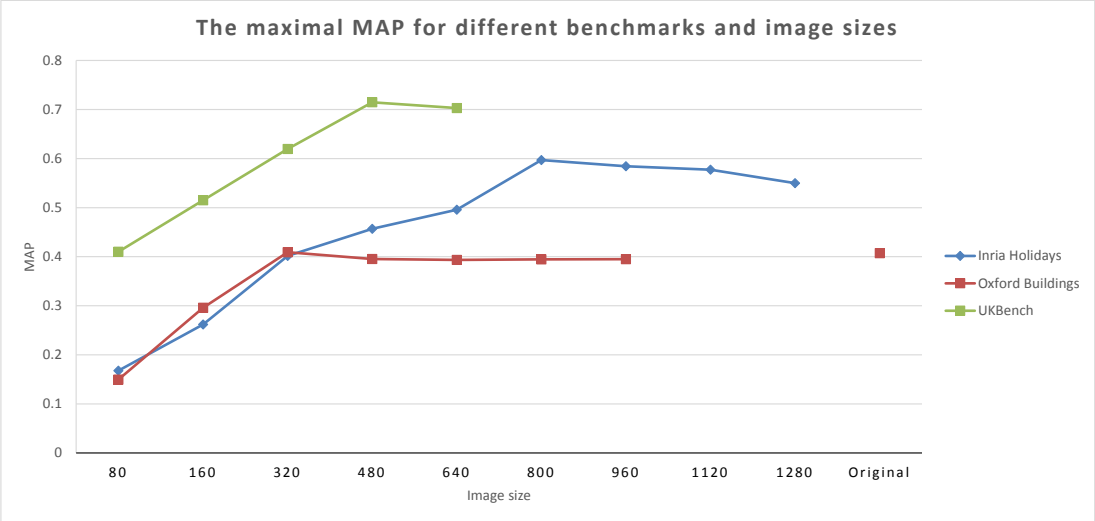


Figure 4.16: The maximal MAP for various benchmarks for different image sizes.

4.3.3 VLAD size

To get optimal results for each image size, it is necessary to adjust other parameters — the settings used for larger images can easily fail for smaller images. One of the most important parameters of the whole VLAD-based model is the size of the VLAD vector — the number of centroids in the vocabulary.

As first, we focus on the precision we can get for various VLAD sizes. For all high-resolution datasets we can get better results with increasing VLAD size as demonstrated in figure 4.17.

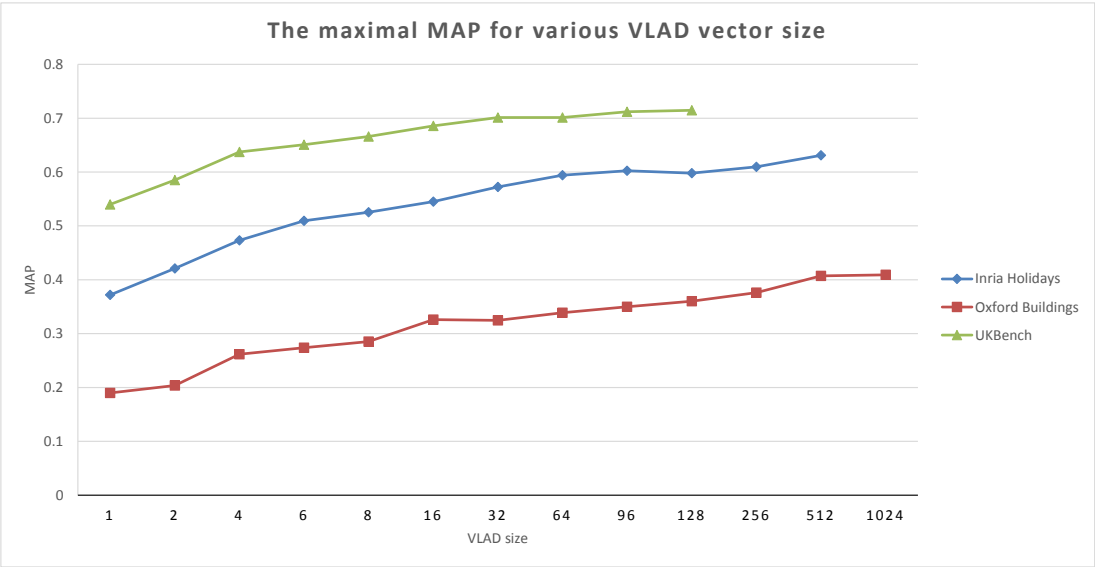


Figure 4.17: The maximal MAP for various benchmarks for different VLAD sizes.

However, it is not true, that for all image sizes, the larger is the VLAD vector, the higher is the precision — for images in lower resolution the excessive enlarging

of the VLAD vector can be very counterproductive.

In figures 4.18 and 4.20 we can see, that in the Oxford benchmark, larger VLAD vector leads, with a few exceptions, to better precision for all image sizes.

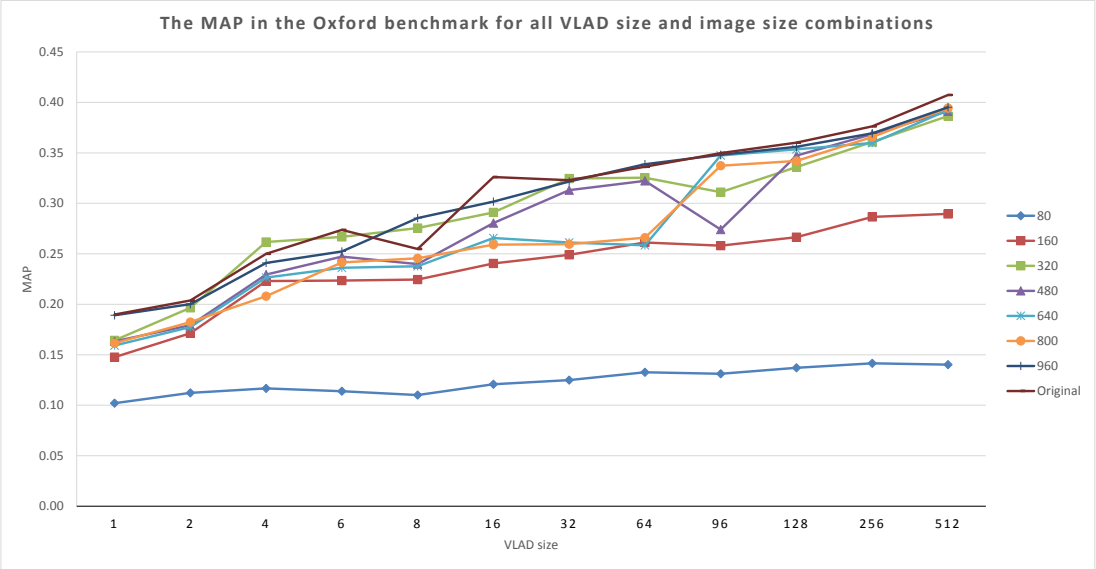


Figure 4.18: The Oxford Buildings benchmark — the impact of the VLAD size on the MAP for various image sizes.

However in the Holidays dataset the situation is quite different, as can be seen in figure 4.19. The same statement applies just for 800px images and larger. For smaller images the MAP decreases a lot for larger VLAD sizes.

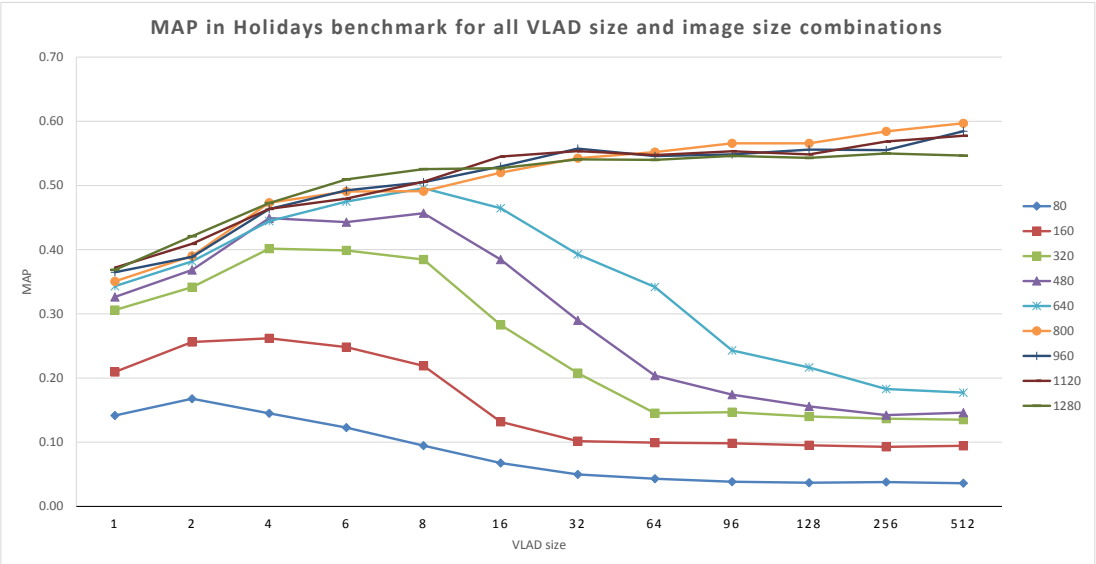


Figure 4.19: Holidays benchmark — The impact of the VLAD size on the MAP for various image sizes.

In figure 4.20 we can see the values of the MAP. In the Oxford dataset there are no big differences when the image or the VLAD size is changed one level up

or down. But in the Holidays dataset, we can see a big difference between the precision for 640px and 800px images for VLADs of size 32 and larger.

VLAD size \ Image size	80	160	320	480	640	800	960	Original
1	0.10	0.15	0.16	0.16	0.16	0.16	0.19	0.19
2	0.11	0.17	0.20	0.18	0.18	0.18	0.20	0.20
4	0.12	0.22	0.26	0.23	0.23	0.21	0.24	0.25
6	0.11	0.22	0.27	0.25	0.24	0.24	0.25	0.27
8	0.11	0.22	0.28	0.24	0.24	0.25	0.29	0.25
16	0.12	0.24	0.29	0.28	0.27	0.26	0.30	0.33
32	0.13	0.25	0.32	0.31	0.26	0.26	0.32	0.32
64	0.13	0.26	0.33	0.32	0.26	0.27	0.34	0.34
96	0.13	0.26	0.31	0.27	0.35	0.34	0.35	0.35
128	0.14	0.27	0.34	0.35	0.35	0.34	0.36	0.36
256	0.14	0.29	0.36	0.37	0.36	0.37	0.37	0.38
512	0.14	0.29	0.39	0.39	0.39	0.39	0.40	0.41

VLAD size \ Image size	80	160	320	480	640	800	960	1120	1280
1	0.14	0.21	0.31	0.33	0.34	0.35	0.36	0.37	0.37
2	0.17	0.26	0.34	0.37	0.38	0.39	0.39	0.41	0.42
4	0.14	0.26	0.40	0.45	0.44	0.47	0.46	0.46	0.47
6	0.12	0.25	0.40	0.44	0.47	0.49	0.49	0.48	0.51
8	0.09	0.22	0.38	0.46	0.50	0.49	0.50	0.51	0.53
16	0.07	0.13	0.28	0.38	0.46	0.52	0.53	0.55	0.53
32	0.05	0.10	0.21	0.29	0.39	0.54	0.56	0.55	0.54
64	0.04	0.10	0.15	0.20	0.34	0.55	0.55	0.55	0.54
96	0.04	0.10	0.15	0.17	0.24	0.57	0.55	0.55	0.55
128	0.04	0.10	0.14	0.16	0.22	0.57	0.56	0.55	0.54
256	0.04	0.09	0.14	0.14	0.18	0.58	0.56	0.57	0.55
512	0.04	0.09	0.14	0.15	0.18	0.60	0.58	0.58	0.55

Figure 4.20: The values of the MAP for various image and the VLAD sizes. Data for the Oxford benchmark are displayed in the table on the left, the right table shows values for the Holidays benchmark.

Large VLADs with Small Images Abnormality

In the previous section we have shown a big difference of the precision for the Holidays dataset when 640px and 800px images were used. We tried to examine this abnormality and to find the reason why there is such a difference between these image sizes.

First of all we found out that in UKBench, third dataset which provides mid-resolution images, we can observe similar pattern. In figure 4.21 we can see that there is also significant difference between precisions of images with size 320px and 480px for larger VLAD vectors.

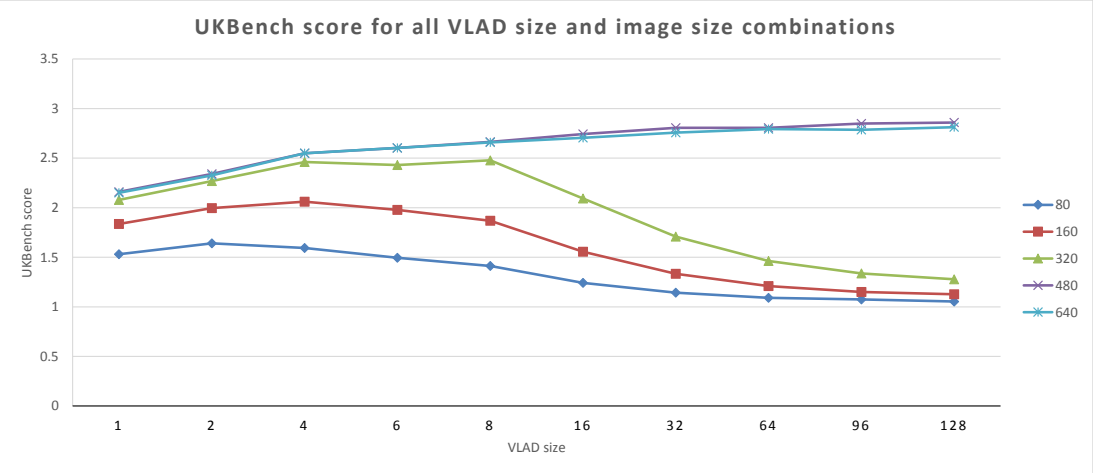


Figure 4.21: The UKBench benchmark — The impact of the VLAD size on UKBench score for various image sizes.

In the second step, we added additional image size classes to the Holidays dataset — 680px, 720px and 760px images. We wanted to know if we take a larger granularity of the image sizes, the precisions will spread equally in the

interval of the precisions that we got for 640px and 800px images or if there will be some breaking image size where there is a significant growth of precision. As can be seen in figure 4.22 the breaking image size is between 680px and 720px images.

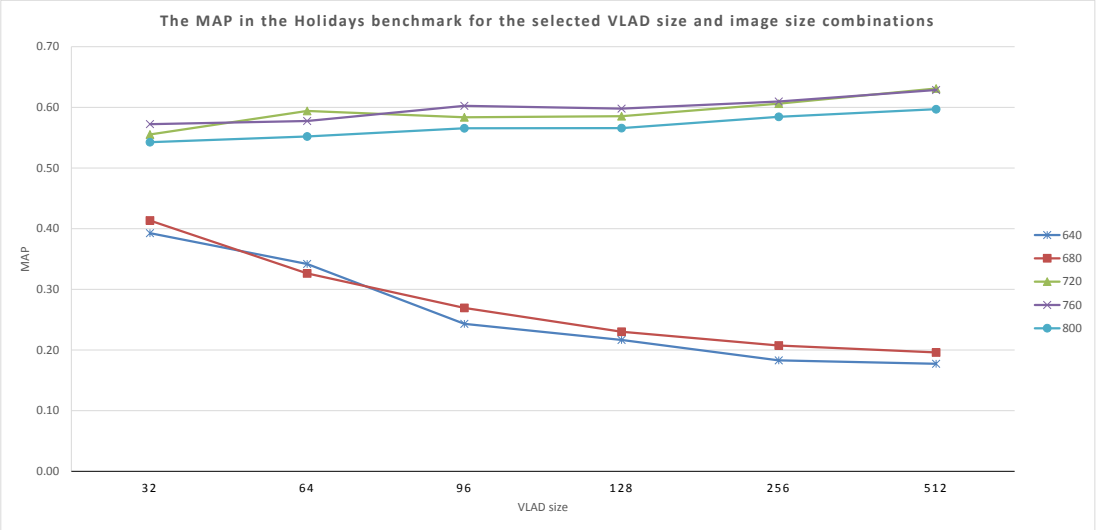


Figure 4.22: The Holidays benchmark — the impact of the VLAD size on the MAP for image sizes between 640px and 800px.

We also focused on the number of the extracted SIFT features, in figure 4.23 we can see that the average count of the SIFT features per one image increases linearly, so there is no correlation between the precision and the number of the extracted SIFT features.

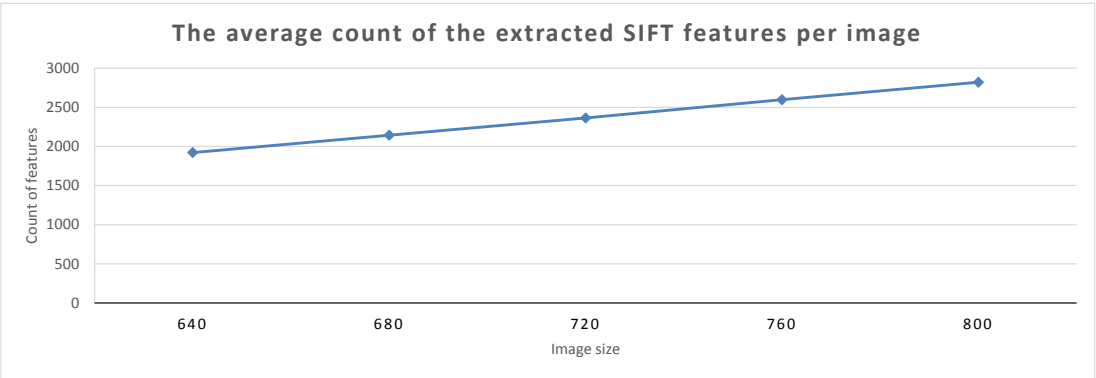


Figure 4.23: The Holidays benchmark — the average count of the extracted SIFT descriptors per one image.

4.3.4 Residual Normalization

One of the possible improvements of the VLAD model presented by Delhumeau et al. [28] is the residual normalization. In figure 4.24 we can see measured MAPs in the Holidays benchmark when residual normalization is used and when it is not. To keep the chart readable, we selected only 3 VLAD sizes. For images smaller than 800px we can see that the residual normalization has negative effect

on the precision, however for 800px images and larger it adds a few percent to the MAP value achieved without residual normalization.

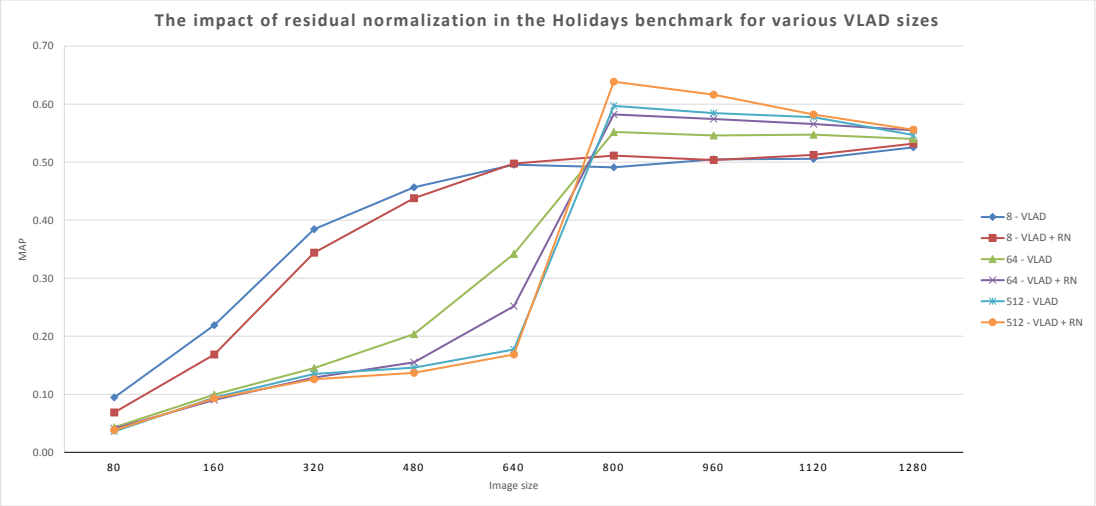


Figure 4.24: The Holidays dataset, MAPs for selected VLAD sizes and all image sizes, with and without residual normalization.

4.3.5 Low Resolution Images

In figure 4.25 the maximal values of the MAP for various benchmarks for 160px images depending on the size of the VLAD vector are displayed. While for the high-resolution images, the larger is the VLAD vector, the larger is the precision, for three of four low-resolution datasets the precision is decreasing with the growing size of the VLAD vector. For example, the VLAD of size 32 leads to 2.5 times lower precision than the VLAD of size 4 when used for 160px images, but for 800px images from the same dataset the larger VLAD leads to the 20% increase of the precision when compared to the smaller VLAD. The only exception is the Oxford benchmark, where with increasing size of the VLAD vector the precision is also increasing.

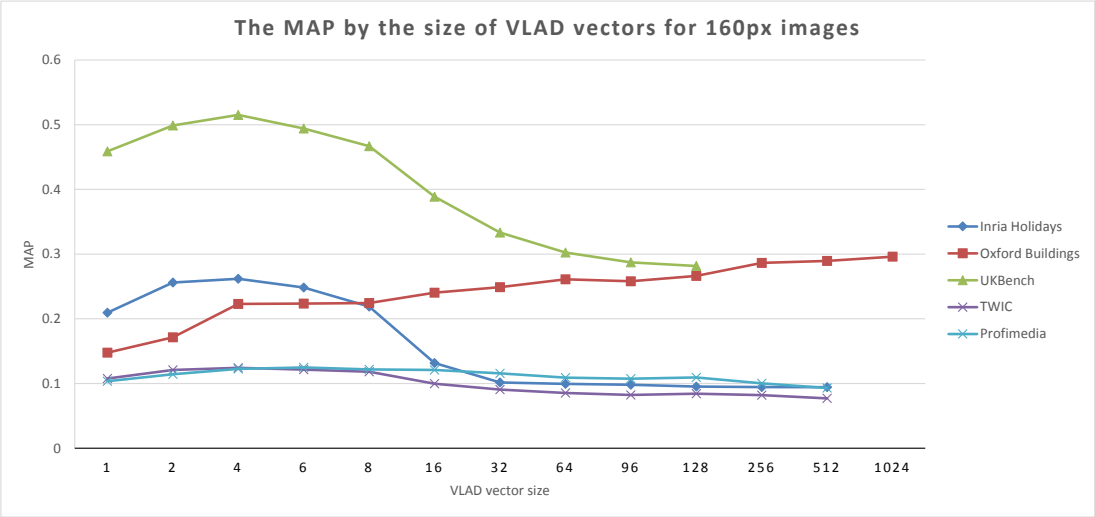


Figure 4.25: Impact of VLAD size on MAP for 160px images and various benchmarks

4.4 VLAD and Signatures Comparison

In the previous sections we have discussed the performance of the VLAD-based and signature-based models in various conditions. In this section we compare the results to find situations where one model significantly outperforms the other.

4.4.1 Low Resolution Images

In figure 4.26 maximal MAP values are presented, for benchmarks with images with the maximal size 160px that were measured by the VLAD-based and signature-based model with the best found configuration. We can see that for low-resolution images the signature-based model significantly outperforms the VLAD model with one exception which is the Oxford benchmark. In section 4.2.4 we have previously shown that the low precision of the signature-based model in the Oxford benchmark is not related to the image size, but to the benchmark itself.

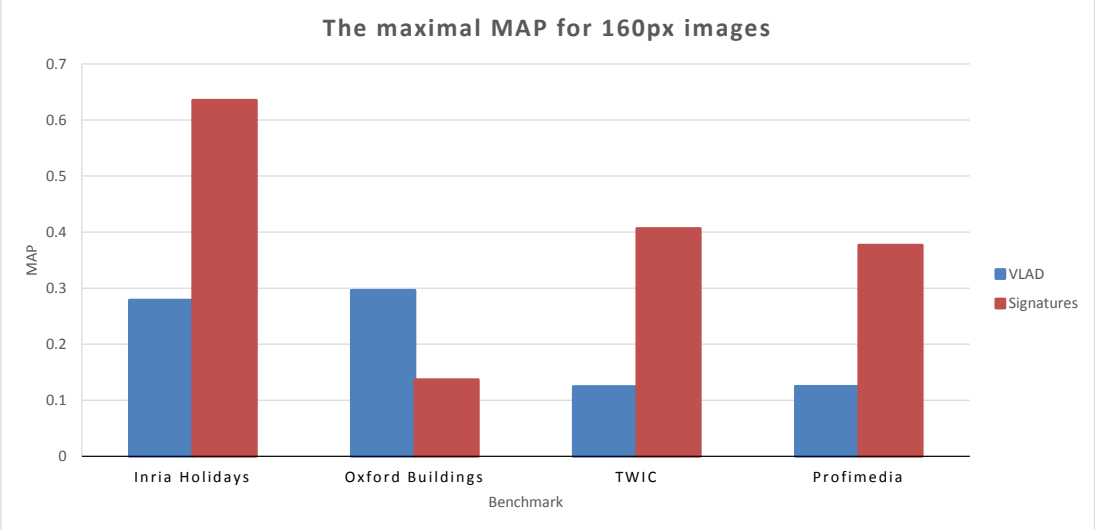


Figure 4.26: The maximal values of the MAP in various benchmarks achieved with the VLAD-based and the signature-based models

4.4.2 Impact of Image Size

In the previous sections we could see that the best values of MAP in the Holidays benchmark, both for the VLAD-based and signature-based models, are approximately the same — 0.638 for the VLAD-based and 0.653 for signature-based. The highest values of MAP for each image size are displayed in figure 4.27. We can see that in the Holidays benchmark for images smaller than 800px the signatures significantly outperforms the VLAD-based models. This fact together with the performance of signature-based models in TWIC and Profimedia low-resolution benchmarks could lead to the statement that for smaller images the signature-based models are better than the VLAD-based models. However, as mentioned before, the results of signature-based models in the Oxford benchmark are very poor, even for the smallest 80px images the results of the VLAD-based models are better, with increasing image size the gap is getting larger.

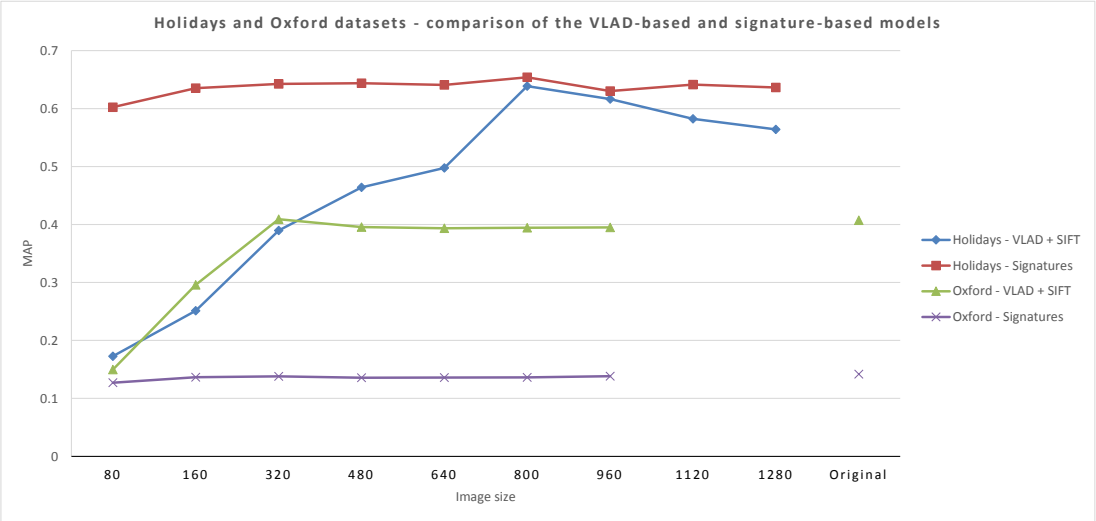


Figure 4.27: The precision of the VLAD-based and signature-based models on the Holidays and Oxford datasets for various image sizes

4.4.3 Precisions of Individual Queries

In the comparison of the signature-based and the VLAD-based models, we also focused on the concrete queries where one of the models failed and the other one had a good precision. In figure 4.28 we can see the average precision for each query both for VLAD-based and signature-based models. If the precisions of both models were similar, the values would be close to the diagonal, but we can see two groups, where one of the models fails ($AP < 0.1$) whereas the second model works well ($AP > 0.9$).

In figure 4.29 we can see the distribution of precisions in a table with values grouped into intervals, in figure 4.28 it is not possible to see the number of points with the same precisions — e.g. in the point with coordinates $(1, 1)$ there are 188 values.

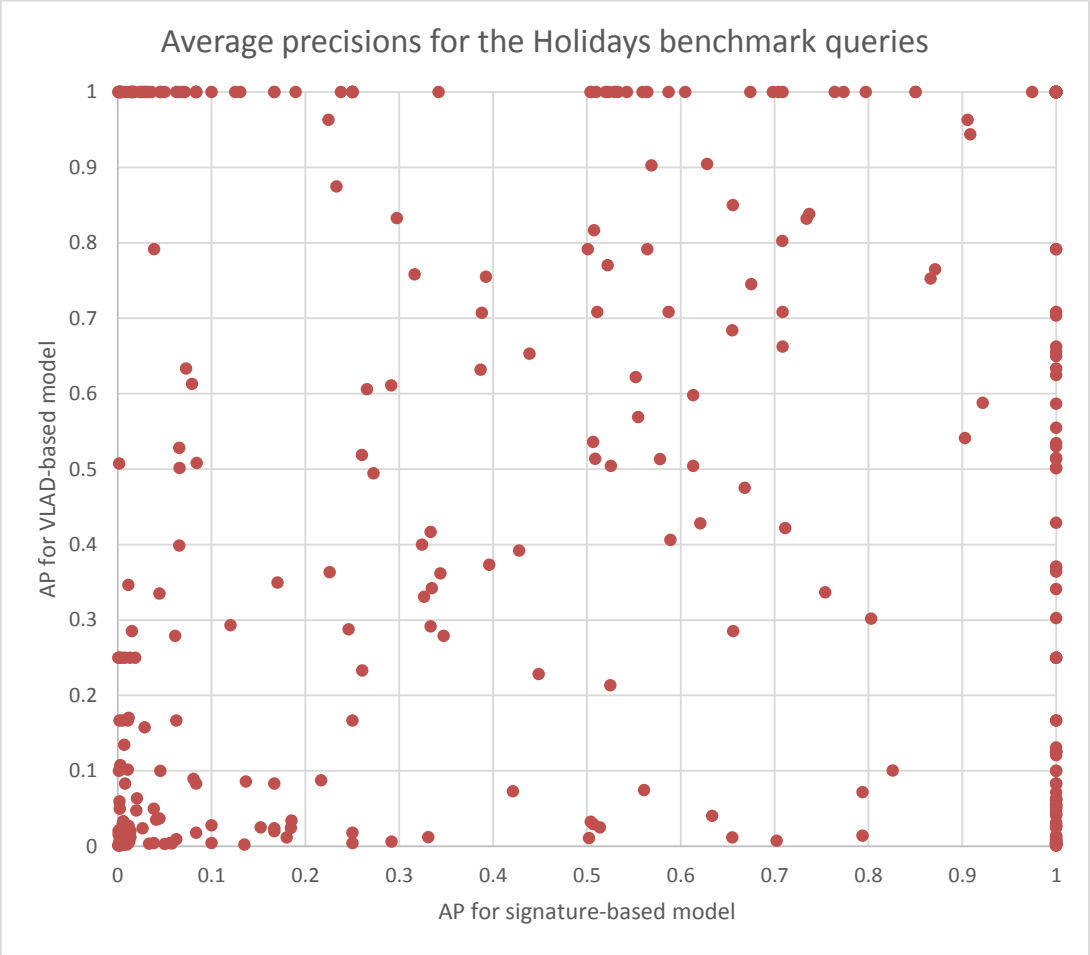


Figure 4.28: Average precisions of all queries in the Holidays dataset measured for the VLAD-based and the signature-based model.

AP for VLAD-based model										
0.9-1	35	6	7	1		12	4	5	2	188
0.8-0.9			2			1	1	3		
0.7-0.8	1			3		5	1	1	2	4
0.6-0.7	2		2	1	1	1	1	1		5
0.5-0.6	4		1			5	2			10
0.4-0.5			1	1		1	2	1		1
0.3-0.4	3	1	1	5	1			1	1	5
0.2-0.3	8	1	2	2	1	1	1			3
0.1-0.2	11		1						1	10
0-0.1	44	11	4	1	1	5	2	3		42
	0-0.1	**0.1-0.2**	**0.2-0.3**	**0.3-0.4**	**0.4-0.5**	**0.5-0.6**	**0.6-0.7**	**0.7-0.8**	**0.8-0.9**	**0.9-1**

AP for siganture-based model

Figure 4.29: Average precisions of all queries in the Holidays dataset measured for the VLAD-based and the signature-based model. Data are grouped into intervals with size 0.1 and each cell contains the number of queries with the average precision belonging to the corresponding intervals.

From the table we can see that for 188 of 500 queries both models work well (the upper right corner) and for 44 queries both models fail (the lower left corner). However there are 77 queries where one of the models fails and the other works well (the upper left and the lower right corner). We can see examples of images where one of the model fails in figures 4.30 and 4.31

If we would be able to choose the correct model for each query and use its result, it would increase the MAP from 0.65 (the MAP of the signature-based model) to 0.79.

Figure 4.30: An example of two pairs of images from the Holidays dataset [29] where the signature-based model fails but the VLAD-based model works well. In both pairs the query image is on the left, the image that should be retrieved is on the right.

Figure 4.31: An example of two pairs of images from the Holidays dataset [29] where the VLAD-based model fails but the signature-based model works well. In both pairs the query image is on the left, the image that should be retrieved is on the right.

Conclusion

In this thesis we have presented numerous measurements where we evaluated the precision of selected CBIR models on the presented datasets. We investigates the behaviour of models in different conditions – how various parameters affect the precision as well as the impact of the image resolution on the precision and optimal settings.

Presented data may be useful for people who build applications based on the presented models as well as for further research in this area.

Another contribution of this thesis is the implementation of benchmarks and other components into the multimedia exploration framework which can be easily reused for future measurements.

Future Work

There are several things that could be processed in the future work:

- As was shown in section 4.3.1, the precision of used extractor of the SURF features is not optimal, it is expected that other extractors, e.g. Modified Upright SURF (MU-SURF) [49] or selected extractors mentioned in the article by Abeles and Peter [48], could have better precision.
- Other state-of-the-art extensions of the BoW model than VLAD could be implemented in the multimedia exploration framework. However it is not expected that the performance will be much different, it would be useful to prove it in all implemented benchmarks.
- New image features called Caffe [50] were introduced recently. According to the published results, these new image features have very high precision on various datasets. It would be useful to have the exact comparison with models presented in this thesis.

Bibliography

[1] Azriel Rosenfeld. "Picture Processing by Computer". In: *ACM Comput. Surv.* 1.3 (Sept. 1969), pp. 147–176.

[2] John Eakins et al. "Content-based Image Retrieval". In: *Library and Information Briefings* 85 (1999), pp. 1–15.

[3] Sagarmay Deb. *Multimedia systems and content-based image retrieval*. IGI Global, 2004.

[4] Kriengkrai Porkaew, Sharad Mehrotra, and Michael Ortega. "Query Reformulation for Content Based Multimedia Retrieval in MARS". In: *Proceedings of the IEEE International Conference on Multimedia Computing and Systems - Volume 2*. ICMCS '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 747–.

[5] Jakub Lokoč et al. "Visual image search: feature signatures or/and global descriptors". In: *Proceedings of the 5th international conference on Similarity Search and Applications*. SISAP'12. Toronto, ON, Canada: Springer-Verlag, 2012, pp. 177–191.

[6] Christian Beecks, Merih Seran Uysal, and Thomas Seidl. "Signature Quadratic Form Distance". In: *Proceedings of the ACM International Conference on Image and Video Retrieval*. CIVR '10. Xi'an, China: ACM, 2010, pp. 438–445.

[7] Thomas Seidl and Hans-Peter Kriegel. "Efficient User-Adaptable Similarity Search in Large Multimedia Databases". In: *Proceedings of the 23rd International Conference on Very Large Data Bases*. VLDB '97. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 506–515.

[8] C. Beecks and J. Lokoč, T. Seidl, and T. Skopal. "Indexing the Signature Quadratic Form Distance for Efficient Content-Based Multimedia Retrieval". In: *Proc. ACM Int. Conf. on Multimedia Retrieval*. 2011, 24:1–24:8.

[9] BoGun Park, KyoungMu Lee, and SangUk Lee. "A New Similarity Measure for Random Signatures: Perceptually Modified Hausdorff Distance". In: *Advanced Concepts for Intelligent Vision Systems*. Ed. by Jacques Blanc-Talon et al. Vol. 4179. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, pp. 990–1001.

[10] Daniel P. Huttenlocher et al. "Comparing Images Using the Hausdorff Distance". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15 (1993), pp. 850–863.

[11] David G. Lowe. "Object Recognition from Local Scale-Invariant Features". In: *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*. ICCV '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 1150–.

[12] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "SURF: Speeded Up Robust Features". English. In: *Computer Vision – ECCV 2006*. Ed. by Aleš Leonardis, Horst Bischof, and Axel Pinz. Vol. 3951. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, pp. 404–417.

[13] Martin Kruliš, Jakub Lokoč, and Tomáš Skopal. "Efficient Extraction of Feature Signatures Using Multi-GPU Architecture". English. In: *Advances in Multimedia Modeling*. Ed. by Shipeng Li et al. Vol. 7733. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pp. 446–456.

[14] T. Sikora. "The MPEG-7 visual standard for content description-an overview". In: *Circuits and Systems for Video Technology, IEEE Transactions on* 11.6 (June 2001), pp. 696–702.

[15] A.E. Abdel-Hakim and A.A. Farag. "CSIFT: A SIFT Descriptor with Color Invariant Characteristics". In: *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. Vol. 2. 2006, pp. 1978–1983.

[16] Yan Ke and R. Sukthankar. "PCA-SIFT: a more distinctive representation for local image descriptors". In: *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*. Vol. 2. June 2004,

[17] Eric N. Mortensen, Hongli Deng, and Linda Shapiro. "A SIFT Descriptor with Global Context". In: *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*. CVPR '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 184–190.

[18] J. Wu et al. "A Comparative Study of SIFT and its Variants". In: *Measurement Science Review* 13.3 (June 2013), pp. 122–131.

[19] James M. Kasson and Wil Plouffe. "An Analysis of Selected Computer Interchange Color Spaces". In: *ACM Trans. Graph.* 11.4 (Oct. 1992), pp. 373–405.

[20] Stephen O'Hara and Bruce A. Draper. "Introduction to the Bag of Features Paradigm for Image Classification and Retrieval". In: *CoRR* abs/1101.3354 (2011).

[21] G. Salton, A. Wong, and C. S. Yang. "A Vector Space Model for Automatic Indexing". In: *Commun. ACM* 18.11 (Nov. 1975), pp. 613–620.

[22] Tapas Kanungo et al. *An Efficient k-Means Clustering Algorithm: Analysis and Implementation*. 2000.

[23] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.

[24] Herve Jegou et al. "Aggregating Local Image Descriptors into Compact Codes". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 34.9 (Sept. 2012), pp. 1704–1716.

[25] Relja Arandjelovic and Andrew Zisserman. "All About VLAD". In: *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*. CVPR '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 1578–1585.

[26] Florent Perronnin and Christopher R. Dance. “Fisher Kernels on Visual Vocabularies for Image Categorization”. In: *2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007), 18-23 June 2007, Minneapolis, Minnesota, USA*. 2007.

[27] I.T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer, 2002.

[28] Jonathan Delhumeau et al. “Revisiting the VLAD Image Representation”. In: *Proceedings of the 21st ACM International Conference on Multimedia*. MM ’13. Barcelona, Spain: ACM, 2013, pp. 653–656.

[29] Herve Jegou, Matthijs Douze, and Cordelia Schmid. “Hamming Embedding and Weak Geometric Consistency for Large Scale Image Search”. In: *Proceedings of the 10th European Conference on Computer Vision: Part I*. ECCV ’08. Marseille, France: Springer-Verlag, 2008, pp. 304–317.

[30] Jakub Lokoč. “Tree-based Indexing Methods for Similarity Search in Metric and Nonmetric Spaces”. In: *Doctoral Thesis, Faculty of Mathematics and Physics, Charles University in Prague*. 2010.

[31] J. Philbin et al. “Object Retrieval with Large Vocabularies and Fast Spatial Matching”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2007.

[32] Relja Arandjelović James Philbin and Andrew Zisserman. *The Oxford Buildings Dataset*. `http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/`. Accessed: 2015-03-22.

[33] Herve Jegou and Matthijs Douze. *INRIA Holidays dataset*. `http://lear.inrialpes.fr/people/jegou/data.php`. Accessed: 2015-03-22.

[34] David Nistér and Henrik Stewénius. “Scalable Recognition with a Vocabulary Tree”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 2. June 2006, pp. 2161–2168.

[35] Henrik Stewénius and David Nistér. *The University of Kentucky Recognition Benchmark*. `http://www.vis.uky.edu/~stewe/ukbench/`. Accessed: 2015-03-22.

[36] Petra Budikova, Michal Batko, and Pavel Zezula. “Evaluation platform for content-based image retrieval systems”. In: *Proceedings of the 15th international conference on Theory and practice of digital libraries: research and advanced technology for digital libraries*. TPDL’11. Berlin, Germany: Springer-Verlag, 2011, pp. 130–142.

[37] Tomáš Grošup et al. “A Web Portal for Effective Multi-model Exploration”. English. In: *MultiMedia Modeling*. Ed. by Xiangjian He et al. Vol. 8936. Lecture Notes in Computer Science. Springer International Publishing, 2015, pp. 315–318.

[38] Tomáš Grošup. “Multi-model Approach For Effective Multimedia Exploration”. In: *Master Thesis, Faculty of Mathematics and Physics, Charles University in Prague.* 2014.

[39] Přemysl Čech. “Using Metric Indexes For Effective and Efficient Multimedia Exploration”. In: *Master Thesis, Faculty of Mathematics and Physics, Charles University in Prague.* 2014.

[40] Martin Fowler. *Inversion of Control Containers and the Dependency Injection pattern*. http://martinfowler.com/articles/injection.html. Accessed: 2015-07-24.

[41] *Autofac: An addictive .NET IoC container*. http://autofac.org/. Accessed: 2015-07-24.

[42] *Emgu CV - cross platform .Net wrapper for OpenCV*. http://www.emgu.com/. Accessed: 2015-04-11.

[43] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[44] *NUnit - a unit-testing framework for all .Net languages*. http://www.nunit.org/. Accessed: 2015-07-25.

[45] *EmguCV documentation*. http://www.emgu.com/wiki/files/2.4.10/document/index.html. Accessed: 2015-03-29.

[46] E. Spyromitros-Xioufis et al. "An empirical study on the combination of surf features with VLAD vectors for image search". In: *Image Analysis for Multimedia Interactive Services (WIAMIS), 2012 13th International Workshop on*. May 2012, pp. 1–4.

[47] *BoofCV - an open source Java library for real-time computer vision and robotics applications*. http://boofcv.org/. Accessed: 2015-07-25.

[48] Peter Abeles. "Speeding Up SURF". English. In: *Advances in Visual Computing*. Ed. by George Bebis et al. Vol. 8034. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pp. 454–464.

[49] Motilal Agrawal, Kurt Konolige, and MortenRufus Blas. "CenSurE: Center Surround Extremas for Realtime Feature Detection and Matching". English. In: *Computer Vision – ECCV 2008*. Ed. by David Forsyth, Philip Torr, and Andrew Zisserman. Vol. 5305. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, pp. 102–115.

[50] Yangqing Jia et al. "Caffe: Convolutional Architecture for Fast Feature Embedding". In: *arXiv preprint arXiv:1408.5093* (2014).

List of Abbreviations

BoW Bag of Words

CBIR Content Based Image Retrieval

DI Dependency Injection

IoC Inversion of Control

IR Image Retrieval

k-NN k-nearest-neighbours

MAP Mean Average Precision

MU-SURF Modified Upright SURF

PCA Principal Component Analysis

PCT Position Colour Texture

PMHD Perceptually Modified Hausdorff Distance

SIFT Scale Invariant Feature Transform

SQFD Signature Quadratic Form Distance

SURF Speeded Up Robust Features

TWIC Thematic Web Images Collection

VLAD Vector of Locally Aggregated Descriptors

Attachments

The following documents or programs can be found on attached DVD:

- PDF version of this thesis
- Stylesheet with all data described in chapter 4
- Source code of the multimedia exploration framework including programs created for the purpose of this thesis described in chapter 3
- Programmer's guide to the multimedia exploration framework
- Programmer's guide to the benchmarks